Desenvolupament d'aplicacions web segures

Treball Final de Grau

Grau en Enginyeria Multimèdia

Cognoms: Garcia Sabaté Nom: Maria del Mar

Pla: 2009

Director: Careglio, Davide

RESUM

El treball de final de grau que s'ha triat consisteix en un recull de vulnerabilitats en quant a seguretat informàtica que es troben amb més freqüència dins de les aplicacions web.

Aquest treball s'ha escollit partint de la experiència pròpia des d'ambdós punts de vista: com a desenvolupadora i també com a atacant sobre tecnologies web. La primera ha sigut necessària per tal de veure quins aspectes les empreses o particulars acostumen a ignorar a l'hora de exposar creacions web; la segona ha consistit en la recerca de buits dintre de la lògica d'aquestes aplicacions web les quals, a través de mecanismes nous o coneguts, han sigut explotats per a benefici d'un tercer no autoritzat.

Per tal de que tothom que es trobi dins de l'àmbit de creació web pugui seguir fàcilment les explicacions, s'ha procurat una dialèctica enfocada dins del món de la creació digital i introduint a mesura que s'explica els mecanismes d'explotació web que tals creadors digitals haurien de conèixer per tal de desenvolupar continguts més segurs.

S'exposarà també la construcció de la pàgina web amb la tecnologia Flask i de quina manera s'ha procurat fer-la lo suficientment segura per trobar-se exposada a Internet, així com també els motius pels quals s'ha escollit aquesta tecnologia. Es farà una breu explicació de les vulnerabilitats exposades a mesura que, posant com a exemple la pròpia pàgina web, es vagi explicant de quina manera s'ha assegurat.

S'ha construït, tanmateix, una pàgina web amb diferents apartats explicant breument cadascuna d'aquestes vulnerabilitats i, a més, incloent petites demostracions exemplars perquè es pugui entendre el funcionament de les vulnerabilitats exposades.

Pàgina: https://www.hackerglossary.com

Paraules clau

Ciberseguretat, vulnerabilitats, educació, tecnologies, aplicacions web, open-source

Índex

Glossari

És un apartat opcional que s'inclou quan el document conté signes, símbols, abreviatures, acrònims o termes que poden no ser compresos fàcilment i ràpidament pels possibles lectors. Es tracta d'una llista en la que es defineixen aquests elements.

1. Introducció

1.1 Motivació i formulació del problema

La problemàtica de aplicacions o pàgines web vulnerables s'incrementa dia a dia, ja que, moltes vegades, no s'acostuma a tenir en compte la seguretat d'aquestes, pensant que 'no passarà' o 'ningú atacarà aquesta pàgina web'.

D'altre forma, la majoria de les vegades que una vulnerabilitat es troba dins d'una pàgina o aplicació web, és un tipus de vulnerabilitat que, seguint unes pautes bàsiques i senzilles de desenvolupament segur, seria molt fàcil de resoldre i evitar la gran majoria d'errors lògics que comporten la violació de la seguretat.

1.2 Objectius generals del TFG

L'objectiu d'aquest treball consisteix en crear una sèrie de pautes que un desenvolupador/a pugui seguir, utilitzant sempre una terminologia a la qual hi estigui acostumat/ada, per tal que no sigui de difícil comprensió i no estigui únicament tancada al nínxol dels qui es dediquen únicament a la ciberseguretat.

Aquestes pautes contindran exemples els quals un desenvolupador/a sigui capaç de reconèixer i adaptar dins el seu àmbit de treball, sense que suposi gran dificultat; eliminant, per tant, la necessitat d'una recerca individual per tal de comprendre i desxifrar les vulnerabilitats que en la pàgina web es pretenen exposar.

1.3 Objectius específics del TFG

- Tecnologia Flask:

  Creació d'una pàgina web amb el framework de Python, Flask.

  Es tracta d'un framework el qual dona molta llibertat per la creació de pàgines web i, a més a més, presenta nombroses llibreries i una gran quantitat de metodologies per evitar vulnerabilitats web.

  Flask és una eina amb la qual ja s'hi té certa experiència, però resultava interessant utilitzar-la per tal de provar com de segura és.

- Implementació d'una Sandbox:

  Es crearà un ambient tancat dintre de la màquina virtual en el servidor. Quan es parla d'una màquina d'ambient tancat, es fa referència a un ambient el qual tindrà el seu propi root, sense permisos d'administrador, restringint els processos o els *syscalls*.

  Tot això es farà amb la intenció de que, si algun usuari aconsegueix entrar a l'ambient de desenvolupament, no tindrà accés a les bases de dades ni la infraestructura que fa funcionar la maquina virtual.

1.4 Abast del projecte

Aquest treball anirà especialment dirigit cap als desenvolupadors web els quals vulguin crear contingut d'una forma segura i fiable. La intenció principal és seguir amb el manteniment de la pàgina web un cop finalitzat aquest projecte, procurant anar afegint continguts a mesura que se'ls requereixi, i facilitant als desenvolupadors la tasca de recerca i comprensió sobre les necessitats de seguretat que un entorn multimèdia necessita avui en dia.

A més, també està dirigit a tots aquells que desitgin saber com s'han implementat tecnologies tals com Flask i Sandbox per un projecte com aquest. S'ha donat una utilitat pràctica a cadascuna d'aquestes tecnologies amb la fi de que més desenvolupadors puguin veure que tals eines resulten ser útils i rentables a llarg termini.

Finalment, també està destinat a les empreses que busquen tenir en compte els impactes econòmics amb que la (in)seguretat de les creacions digitals els hi pot afectar el negoci, tant de forma positiva com de forma negativa.

2. Estat de l'art

2.1 Situació dels negocis dins del món digital

La quantitat de recursos web que s'utilitzen en el dia a dia per diferents persones de diferents parts del món s'ha multiplicat durant els últims vint anys, fent que aquesta mena de recursos ja no juguin un paper secundari en la quotidianitat de la societat ni funcionament global.

És per això que es requereix alguna mena de mecanisme per assegurar-se que, totes les dades que cada dia es generen durant la interacció amb aquests recursos, es mantenen segures des del moment en que s'inicien fins al moment en que es finalitzen.

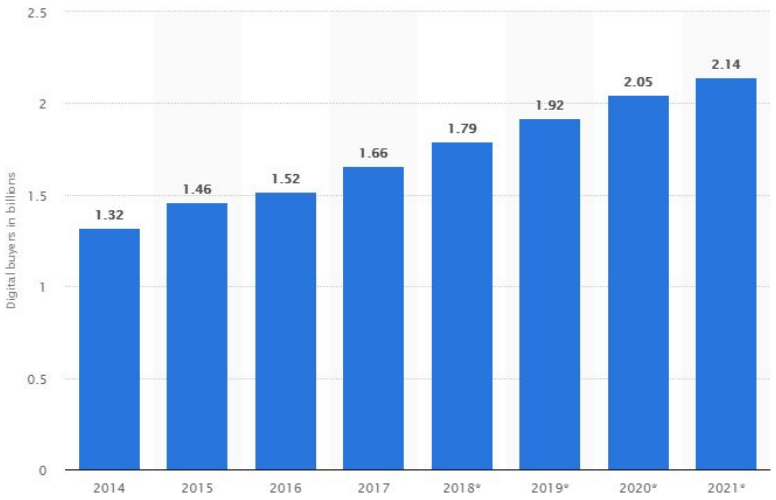
L'any 2014, per exemple, es va saltar d'un nombre de 630 milions de pàgines web a 850 milions al final de l'any [1]. Aquesta quantitat només marca l'inici d'un període en el que, cada vegada amb més freqüència, serà necessari exercir un control sobre les dades que es generen a partir de la interacció entre l'usuari i la pàgina web.

De totes aquestes pàgines web, un gran nombre ja inclouen alguna forma d'intercanvi monetari; més alt encara és el nombre de creacions de comptes dins d'aquestes pàgines web, on s'hi confien dades personals que podrien ser destinades a la monetització, ja fos amb o sense el permís de l'usuari.

La necessitat d'una regulació parteix d'aquesta mateixa base, des d'on un usuari hauria de poder estar segur/a que les seves dades personals no es destinaran per a cap altre fi que pel qual s'ha especificat inicialment abans del registre.

És més preocupant, d'altre banda, la quantitat de transaccions que es duen a terme durant el dia a dia i al llarg de l'any sobre plataformes que no han sigut assegurades.

L'any 2017 es va calcular que el nombre realitzat de transaccions via mètodes oferts per internet arribava als 1.66 mil milions (Il·lustració 1). Es calcula que l'any 2021 la quantitat serà de 2.14 mi milions, i seguirà en escalada[2].



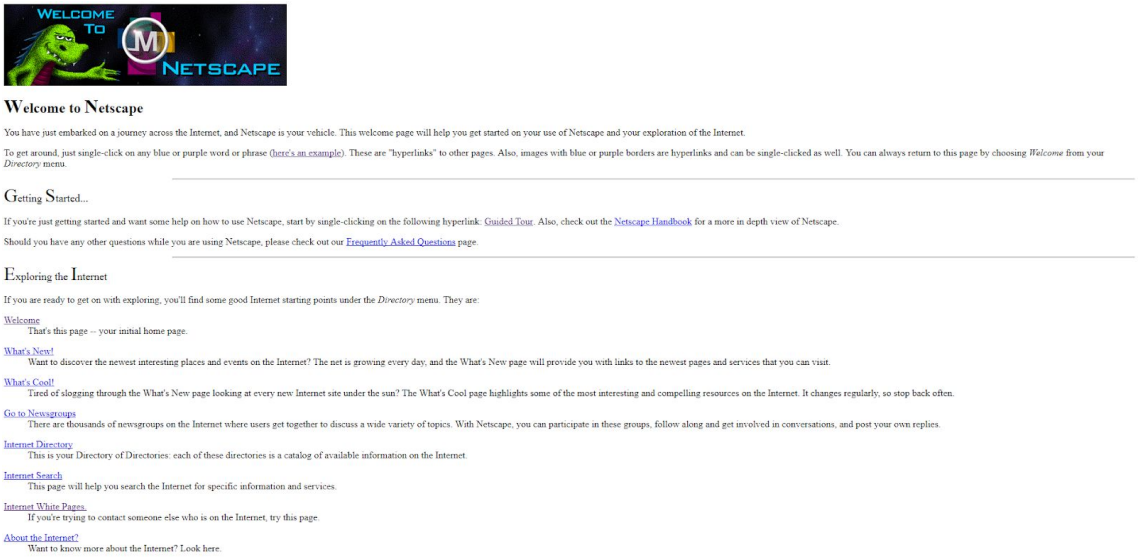
Il·lustració 1 - Compradors online període 2014-2021

No només vol dir això que es comprarà encara més en les grans plataformes que dominen ara el mercat, sinó que cada cop en sorgiran més plataformes que, en certa mesura, oferiran un mecanismes de compra-venta. La estimació de noves pàgines web que es creen cada dia ronda els mil milions, això són entre 300 i 500 pàgines que es creen cada minut.

2.1.1 Inicis: la web com a via de expressió

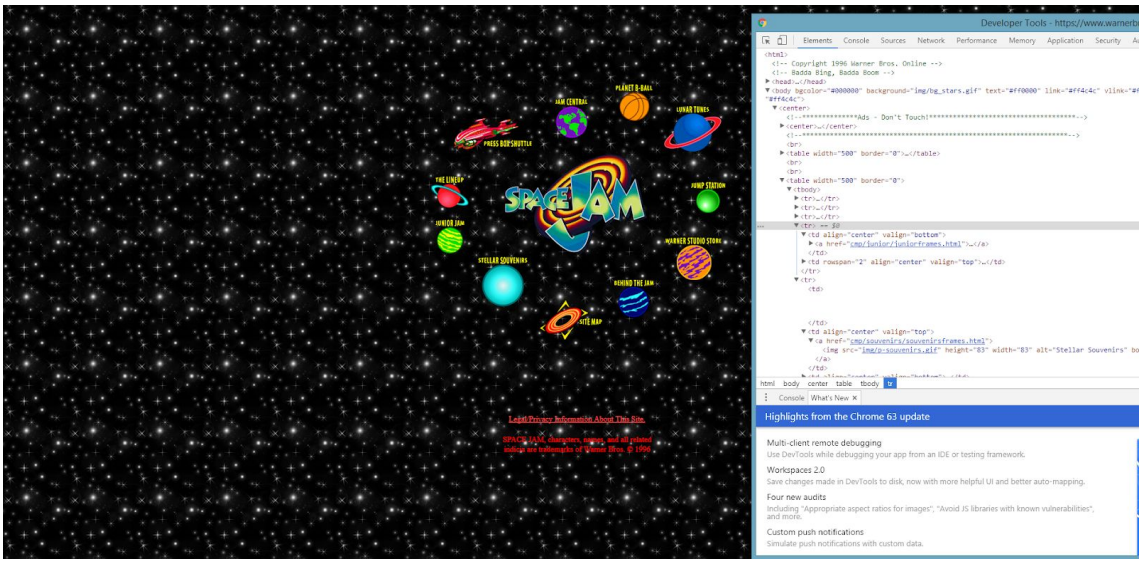
En els seus inicis als anys 90, les pàgines web consistien en vectors per a compartir opinions i punts de vista respecte als diferents temes polítics i/o globals de la època.

Des d'un punt de vista merament tècnic, es tractava de pàgines web molt senzilles, que només feien ús de HTML i CSS per a construir continguts, sense necessitat de compiladors ni abús de JavaScript.



Il·lustració 2 - Pàgina web de Netscape [*Home.mcom.com*]

En la pàgina de *NETSCAPE* podem veure un ús senzill de HTML, el qual inclou imatges i explica com utilitzar els *hyperlinks*.



Il·lustració 3 - Pàgina web de Space Jam

[*https://www.warnerbros.com/archive/spacejam/movie/jam.htm*]

Un altre exemple interessant seria la pàgina web de la pel·lícula SpaceJam (Il·lustració 3), la qual fa ús de la mateixa metodologia que la primera per a mostrar els continguts. Si es mira el codi font de la pàgina web, es pot observar que pertany a l'any 1996.

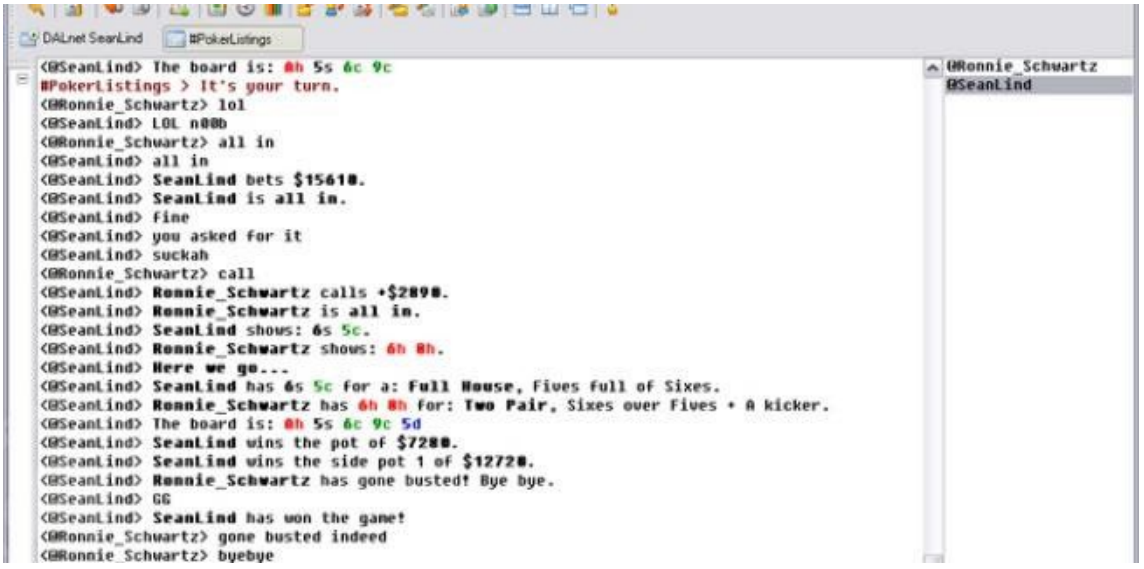
Il·lustració 4 - Altre exemple de pàgina web dels anys 90 [http://toastytech.com/evil/index.html]

Hi ha incomptables exemples de pàgines web fent ús d'aquesta tècnica, la qual va quedar ancorada com a clàssica dels anys 90.

2.1.2 Naixement d'una nova entitat col·lectiva

Alhora que van començar a aparèixer públicament aquestes pàgines web, també va sorgir un nou grup a les xarxes: els primers hackers digitals.

Es tractava de col·lectius de persones individuals, les quals parlaven entre sí mitjançant eines de comunicació com IRC o fòrums especialitzats. La constant disponibilitat d'aquestes formes de comunicació i la globalització d'internet, van donar lloc a un *boom* de la informació, segons el qual, es compartien dades de forma immediata i en qualsevol lloc, donant pas a que qualsevol persona hi pogués contribuir.



[2.1.2 - Imatge del client mIRC per a IRC]

A través d'aquestes eines de comunicació, van aconseguir compartir les troballes que, individualment, cadascun dels membres participants havien trobat a pàgines web o sistemes restringits. La suma de les troballes individuals permetien una ràpida organització dels punts febles, gràcies als quals aconseguien vèncer les restriccions imposades pels creadors/propietaris, i guanyaven accés o privilegis sobre aquests.

2.1.3 Comercialització d'aquesta entitat

A mesura que la informació presentada y guardada a les xarxes creixia, també creixien els mecanismes pels quals aquesta informació podia ésser robada, modificada o destruïda.

Les empreses i/o entitats requerien personal especialitzat en protegir les dades, de forma que no poguessin ser alterades de cap forma. Així doncs, es va començar a introduir una nova posició de treball, la qual es dedicava únicament a enfortir les defenses i aplicar mecanismes de deteccions de intrusions.

2.2 Estimació de pèrdues i guanys econòmics causats per atacs contra les plataformes

La pàgina Bugcrowd [3], la qual es centra en oferir a creadors i atacants un mètode d'ajuda mútua mitjançant una retribució econòmica per part dels creadors cap als atacants si aquests últims troben vulnerabilitats en els seus treballs, va fer un recull de informació sobre les vulnerabilitats que es varen trobar al llarg de l'any en diferents plataformes comercials.

L'any 2017 es va constatar que se'n havien enviat 100000, resultant vàlides unes només 50000 d'aquestes (amb impactes greus).

Il·lustració 5 - Submissions de vulnerabilitats 2013-2017
[https://pages.bugcrowd.com/hubfs/Bugcrowd-2017-State-of-Bug-Bounty-Report.pdf]

Aquest és un nombre que anirà pujant dramàticament al llarg dels anys, doncs de la mateixa forma, la plataforma Bugcrowd va anunciar que la quantitat de programes que es van trobar amb una vulnerabilitat crítica durant les primeres 24 hores va augmentar un 20% durant l'any 2017, una xifra que no sembla que vagi a parar de créixer.

La relació entre la senzillesa del atac i la seva criticitat és directament proporcional, ja que com més fàcil de reproduir sigui, més risc existeix de que una gran quantitat de usuaris l'hagin reproduït. La reproducció d'atacs no autoritzats sobre una plataforma digital comporta la pèrdua de reputació i informació crítica en una pàgina web, resultant, de forma inevitable, en una pèrdua econòmica.

Ocasionalment, aquesta pèrdua de reputació, pèrdua econòmica o filtratge de informació comporta directament el fi de l'existència d'aquesta plataforma o companyia.

2.2.1 Revisió dels atacs amb més impacte des de l'any 2000 i totes les pèrdues generades

Començant l'any 2000, es va fer públic un malware anomenat ILOVEYOU, el qual funcionava mitjançant el servei de email, al qual se li adjuntava un fitxer que, al executar-lo, infectava fitxers de l'ordinador i es propagava a tots els contactes de l'email de l'usuari infectat [4].

Aquest malware va tenir uns perjudicis de 10 bilions de dòlars americans. Els recursos econòmics es van emplear, en gran mesura, a netejar els rastres del malware i recuperar documents.

Un atac més recent, és el que va tindre lloc a l'any 2016, sota el nom de Mirai [5]. Aquest atac va inflitrar-se a dispositius IoT (Internet of Things), oberts a l'internet amb les credencials per defecte.

L'atac va apoderar-se de tots els dispositius per a llençar un atac DDOS cap a pàgines amb un gran pes en el món de les xarxes socials, tals com Twitter, PayPal, Amazon o Netflix.

S'estima que les pèrdues econòmiques van arribar als 110 milions de dòlars, entre temps de no funcionament, reparar danys als sistemes i recuperació de dades.

Aquests dos exemples ens mostra que, tot i que la situació actual per al control de d'atacs, segueixen produint-se usualment per causes com falta de seguretat com verificar la validesa d'un fitxer o bé deixar les credencials per defecte de dispositius connectats a internet.

Així doncs, es confirma que, la majoria dels atacs són causats per atacs de fàcil reproducció i execució, sent aquests els que generen danys en una escala massiva.

2.3 Acolliment dels riscos dins del món del desenvolupament

Per tal de combatre els riscos associats a obrir una plataforma a l'internet, així com la introducció de vulnerabilitats de fàcil reproducció i execució, es va començar a incorporar metodologies de seguretat de la informació dins dels dissenys dels marcs de desenvolupament o, dit d'altre forma, els *frameworks*.

Fins ara, la gran majoria de desenvolupadors confien en la capacitat dels creadors de *frameworks* de desenvolupament per trobar i arreglar els errors de seguretat que mostren les pàgines amb les quals es creen. Aquesta confiança cega pot arribar a generar vulnerabilitats de fàcil explotació, les quals serien evitables amb pautes clares que es pogués seguir.

Sorgeix, d'aquesta forma, la necessitat que trobar mètodes per vincular de nou als creadors amb la seguretat de les seves creacions, i evitar per tant que es creïn aquests buits dintre de la lògica de forma tant extensa.

2.3.1 Introducció de les *frameworks* de treball

Un *framework* es defineix com un marc de desenvolupament de treball. En el món del internet, un *web framework* seria un marc de desenvolupament de pàgines web. Aquests *frameworks* incorporen funcions pròpies que ajuden a gestionar la informació que es passa des de l'usuari fins a la base de dades, així com també a la inversa. Moltes vegades també inclouen mecanismes per a la protecció d'aquestes dades que viatgen des de l'usuari (*front-end*) fins a la base de dades (*back-end*).

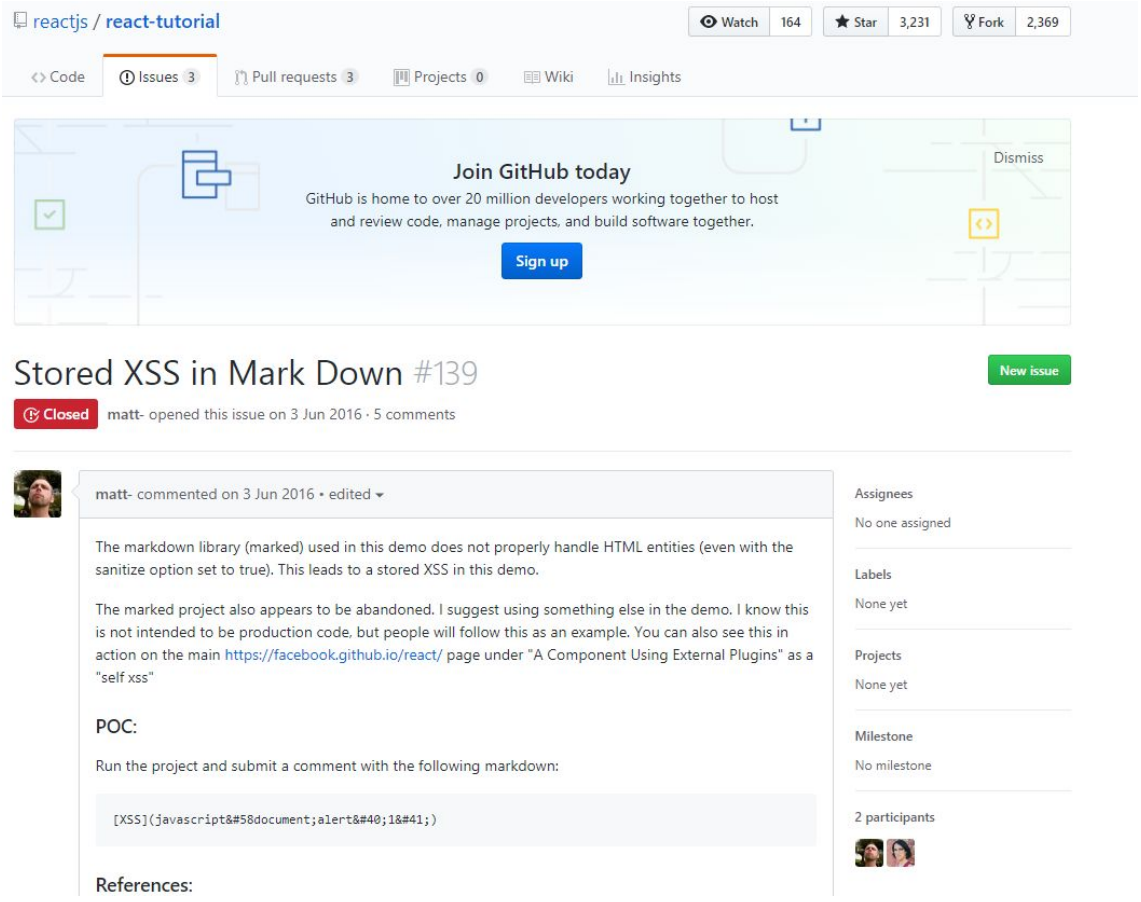
Els *frameworks* ja són àmpliament acollits dins del món del desenvolupament web, ja que tenen la facilitat de tenir la gestió de la informació feta, així com la integració de funcions vitals, com les que gestionen la connexió a la base de dades; o bé ofereixen facilitats per a la creació d'una lògica repetitiva, segons les quals només caldria escriure una vegada, a partir des d'on el *framework* s'encarregaria de repetir la mateixa lògica el nombre de vegades indicat.

Frameworks d'aquest tipus existeixen per a JavaScript (Angular, React), Python (Flask, Django), PHP (Laravel, Symfony), Java (Apache struts), ...

2.3.2 Acolliment de tècniques de seguretat dins aquestes *frameworks*

Per tal de combatre la falta de seguretat dins dels *frameworks*, es van acollir canvis en el seu funcionament. Aquests canvis consistien en dissenys centrats en garantir la seguretat de les dades, tant en la comunicació dels processos interns, com la protecció davant de intents d'atacs externs.

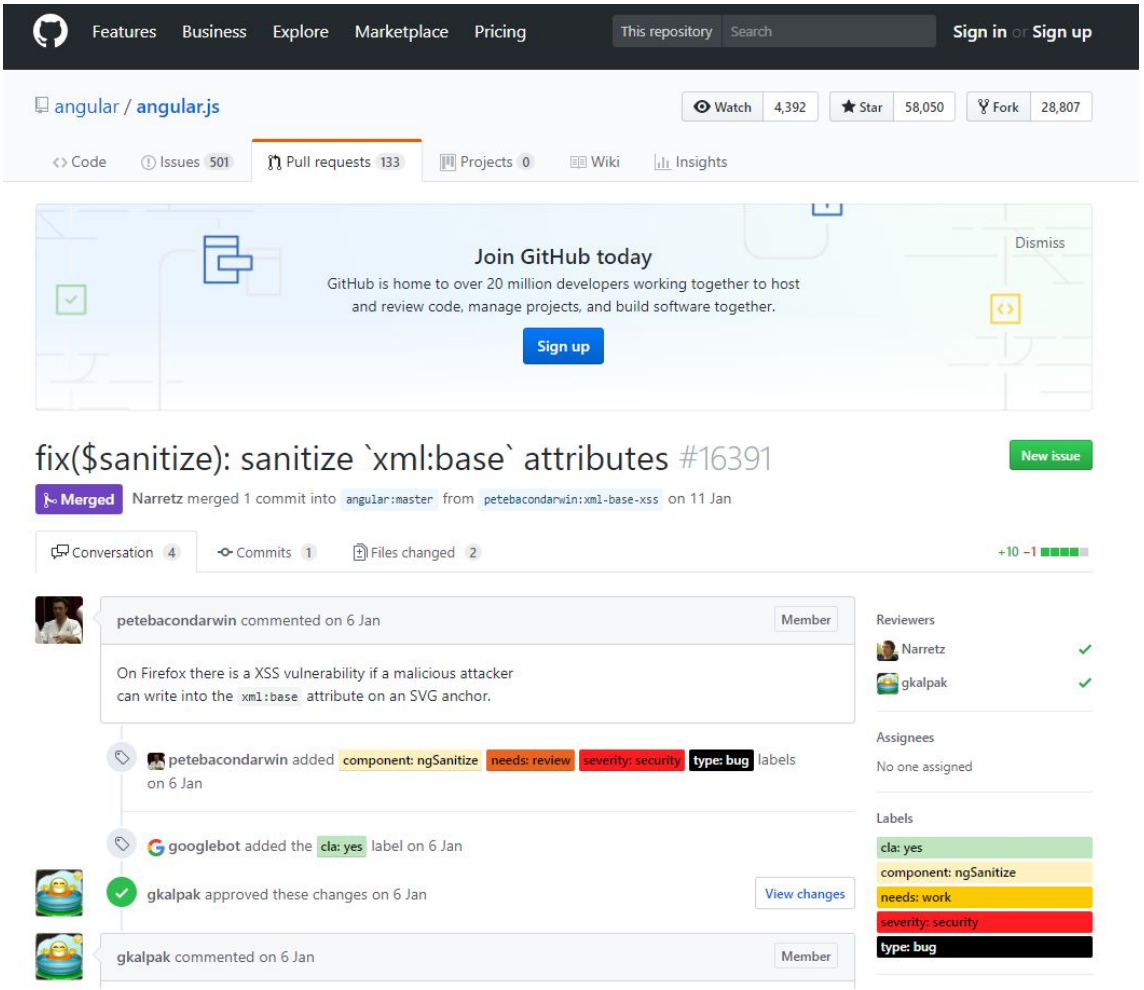
Per exemple, gairebé totes les *frameworks* modernes contenen mecanismes per a la neteja de les dades que un usuari pot introduir.



Il·lustració 6 - Commit per arreglar una vulnerabilitat de ReactJS [*https://github.com/reactjs/react-tutorial/issues/139*]

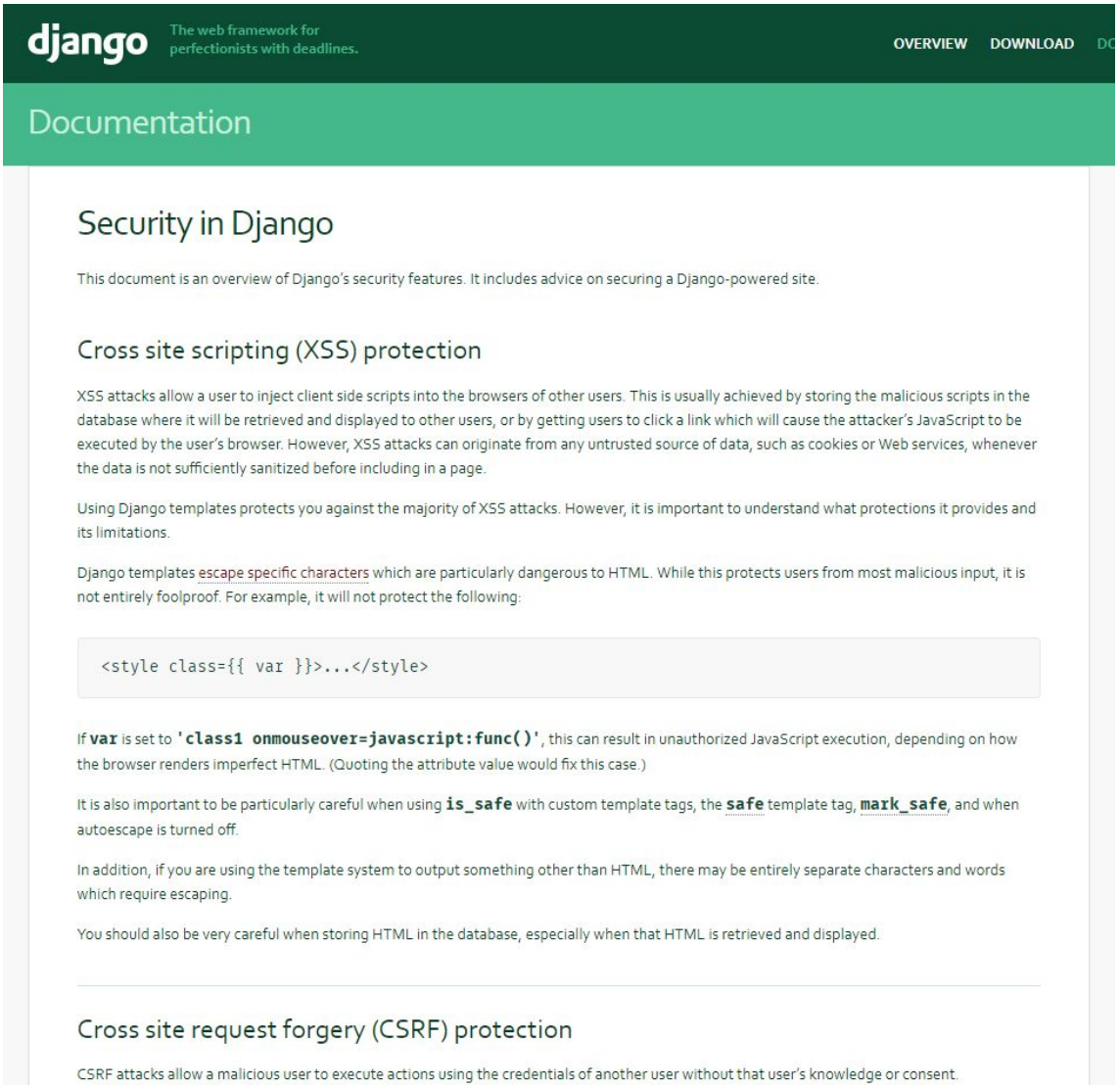
Com per exemple, en el repositori de ReactJS trobem usuaris desenvolupadors que contribueixen a la comunitat de desenvolupadors de ReactJS exposant vulnerabilitats que es troben.

Aquesta és una mesura molt emprada actualment. Fer un projecte *open-source* és un bon mecanisme per a garantir que el projecte pugui créixer i millorar cada dia a mesura que la quantitat de usuaris augmenta.



Il·lustració 7 - Commit per arreglar una vulnerabilitat a AngularJS [https://github.com/angular/angular.js/pull/16391]

D'altra banda, també s'acullen apartats de seguretat dins de les pàgines web d'aquestes *frameworks*, usualment en la documentació del seu funcionament.



Il·lustració 8 - Pàgina amb la documentació de seguretat de Django
[*https://docs.djangoproject.com/en/2.0/topics/security/*]

Tot i la gran quantitat d'aportacions diàries dins del marc de la seguretat de les aplicacions, cada dia en sorgeixen de noves.

Això dona com a resultat la necessitat d'una sèrie de pautes que sigui necessari seguir a l'hora de comprovar que tals aplicacions web segueixen les millors especificacions possibles.

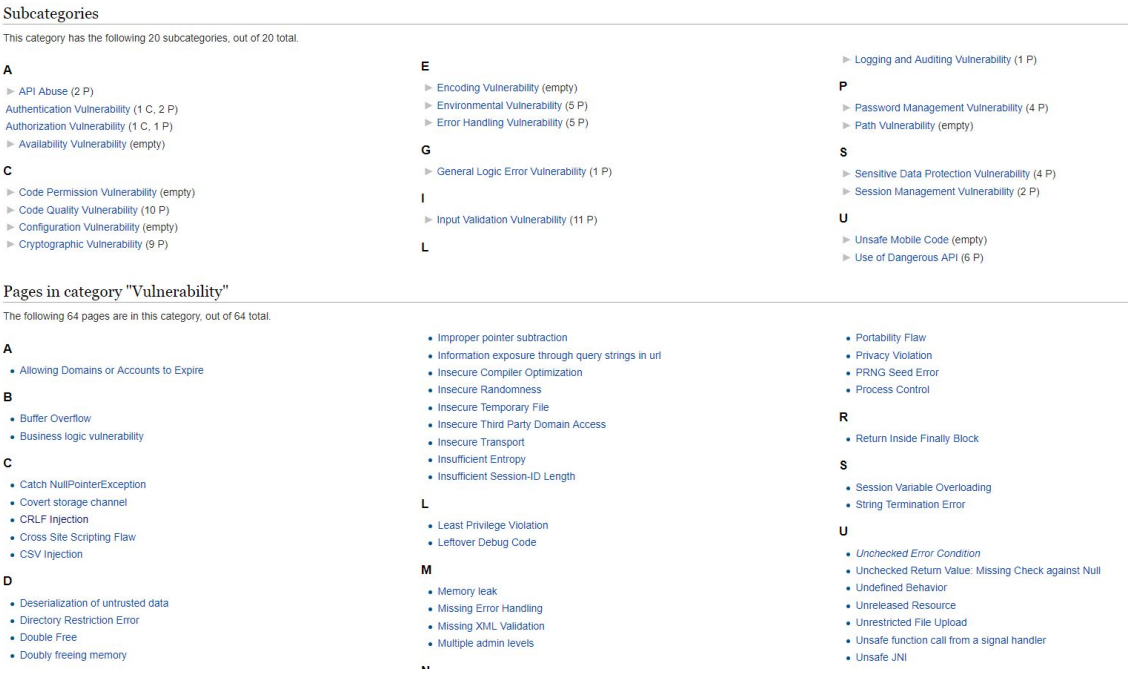
2.4 Previsió d'una necessitat futura de fonts de informació

L'any 2001 va sorgir OWASP [6], una entitat dedicada a fer un seguiment i exposició de les vulnerabilitats més influents i/o amb més repercussió. OWASP es dedica a l'estudi de tals vulnerabilitats, la seva exposició a internet i l'oferiment de mecanismes per tal de pal·liar aquestes vulnerabilitats.

A mesura que sorgeixin cada cop més plataformes digitals, s'introduiran nous vectors d'atac. És, per tant, necessari conscienciar als nous desenvolupadors de la importància de mantenir unes pautes de seguretat.

D'altra banda, existeixen també llocs que expliquen els tipus de vulnerabilitats que hi ha, però moltes vegades no es fa amb la suficient claredat ni està enfocat per a un públic de desenvolupadors, de forma que aquest públic usualment preferirà seguir utilitzant els mecanismes que fins ara ha utilitzat per la creació de pàgines web.

Usualment aquests llocs educatius sobre les diferents vulnerabilitats web existents consisteixen únicament en la teoria, el que causa que es torni una tasca complexa per a persones de diferents nacionalitats o llenguatges, o bé amb qui no està del tot familiaritzat amb la terminologia.

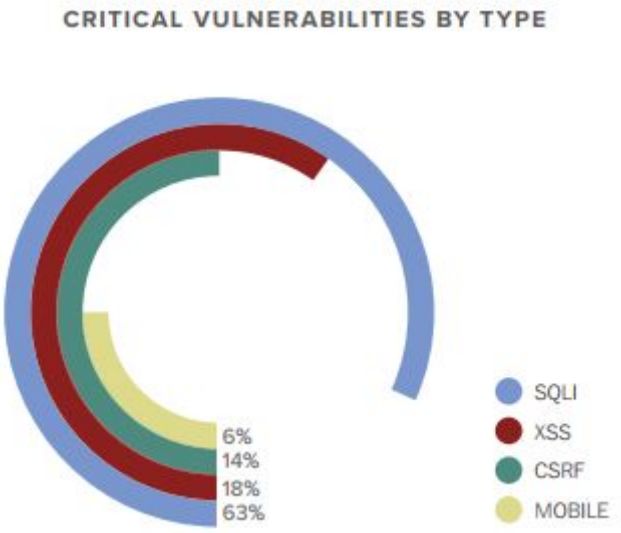


Il·lustració 9 - Índex de vulnerabilitats web de OWASP[https://www.owasp.org/index.php/Category:Vulnerability]

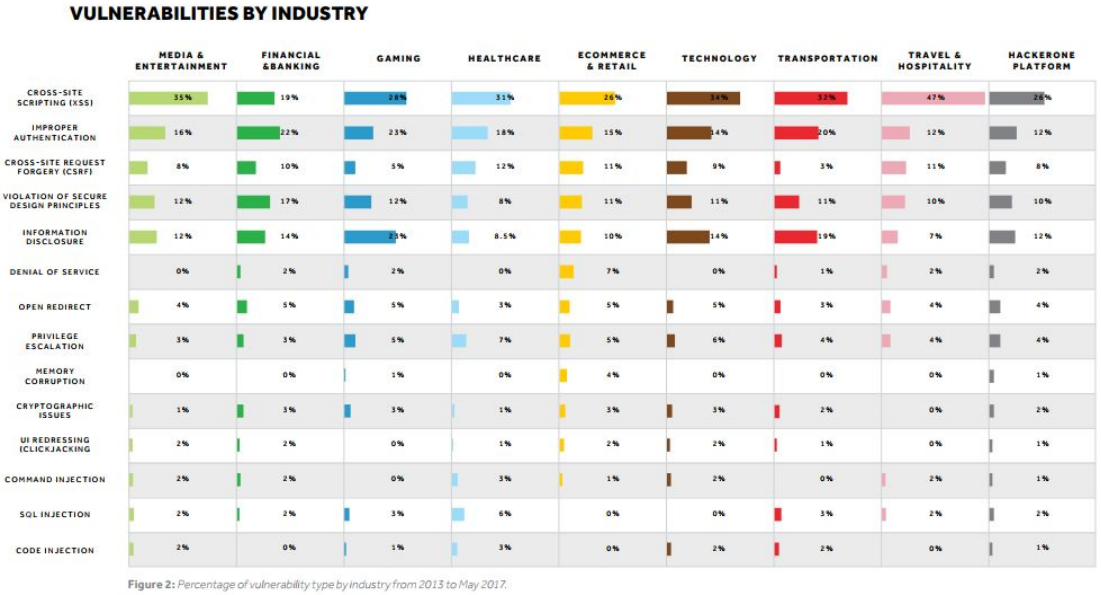
En la imatge a dalt (Il·lustració 9) veiem, per exemple, OWASP categoritzant les vulnerabilitats web. Tot i estar estandarditzat, pot comportar que els desenvolupadors no familiaritzats amb el comportament d'aquestes vulnerabilitats entenguin malament el seu concepte o no ho entenguin del tot.

Aquest treball s'enfocarà, per tant, en la introducció i explicació de les vulnerabilitats que el nínxol de desenvolupadors digitals es podrien trobar dins l'àmbit web. Les vulnerabilitats que es mostren no han sigut pas aleatòries, sinó que s'han escollit partint del OWASP top 10 (el top 10 de

vulnerabilitats més populars durant l'any que registra OWASP) i també les vulnerabilitats més populars que s'han trobat a les plataformes com Bugcrowd o HackerOne.



Il·lustració 10 - Quantitat de vulnerabilitats per tipus
[https://pages.bugcrowd.com/hubfs/Bugcrowd-2017-State-of-Bug-Bounty-Report.pdf]

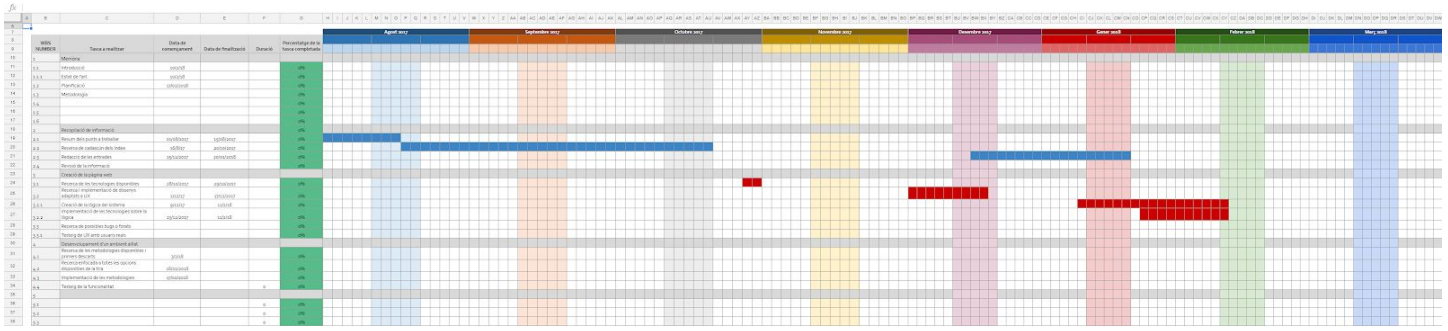


Il·lustració 11 - Quantitat de vulnerabilitats per tipus i indústria
[https://www.hackerone.com/sites/default/files/2017-06/The%20Hacker-Powered%20Security%20Report.pdf]

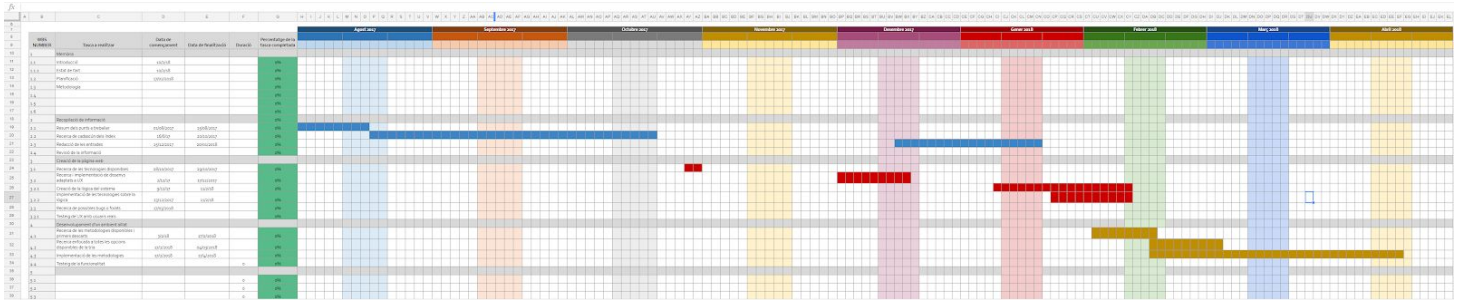
3. Planificació

S'ha realitzat una planificació temporal amb diagrames de Gannt i l'eina Excel per a Google Drive.

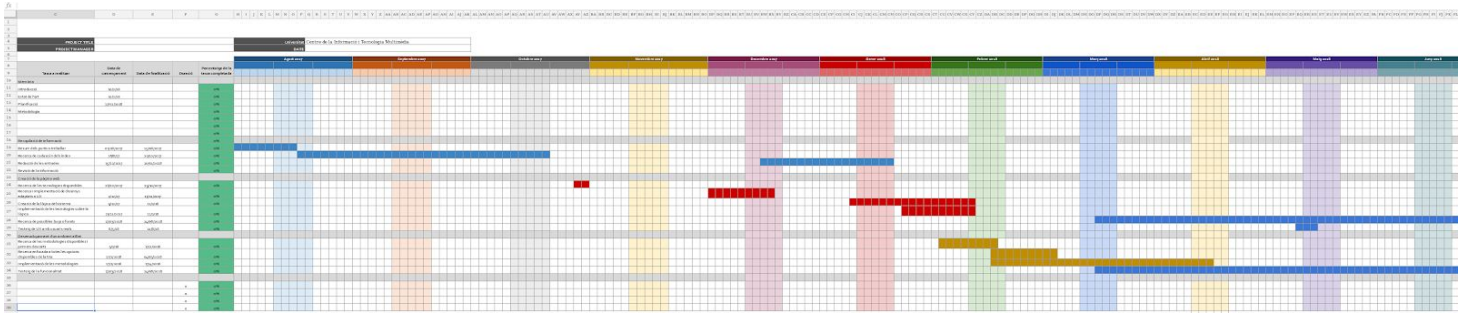
Febrer:



Abril:



Juny:



Gràcies a aquest disseny, s'ha pogut fer un esquema visual de les tasques que s'han realitzat o es realitzaran al llarg del temps, i d'aquesta forma, organitzar-se millor per temps dedicat a cada fase del projecte.

La divisió de les etapes és la següent:

Memòria

Temps dedicat a escriure la documentació necessària per al seguiment i evaluació del projecte.

S'ha dividit els punts dins de la categoria de memòria en els punts que comprenen la descripció de la primera rúbrica. Per això, és una categoria que estarà subjecte a canvis al llarg de la realització de la segona rúbrica i la memòria final.

Recopilació de la informació

Donat el llarg temps que es va requerir per a recopilar tota la informació necessària per a crear cadascun dels apartats de la pàgina web, se li va atorgar la seva pròpia categoria.

Aquest apartat va consistir en una recerca quasi diària de informació en llibres especialitzats i en internet, la seva anotació, la redacció sobre tal i com es presentaria en la pàgina així com la revisió i comprovació de les dades.

Creació de la pàgina web

Es va establir una sèrie de punts al llarg de la creació de la pàgina web. Aquests punts comprenien la elecció d'una tecnologia de desenvolupament després d'una pluja d'idees i d'intents que no van acabar d'agradar.

També inclou el disseny de la lògica interna, la creació i població d'una base de dades (així com la elecció d'aquesta), la correcció d'errors i el disseny basat en UX.

Desenvolupament d'una ambient aïllat

Aquest apartat es va anar desenvolupant al llarg del període entre Abril, Maig i Juny.

A principis d'Abril es va començar l'estudi per la creació d'un ambient aïllat, el qual es va dur a terme durant Maig i Juny.

3.1 Anàlisi DAFO, riscos i pla de contingències

Per a realitzar la planificació dels diferents punts del projecte dins de cadascuna de les seves fases, s'ha procedit a la realització d'un anàlisi DAFO.

En aquest anàlisis s'han considerat diversos aspectes.

Primerament, s'ha tingut en compte la creació de la pàgina web com si fos un negoci, és a dir, creant un pressupost i recreant com seria la situació si surtis al mercat.

La seva posició dins del mercat dependria de la quantitat de projectes similars al que s'està desenvolupant, el col·lectiu que podria fer-ne ús del site i un manteniment constant al qual se li apliquessin millores periòdicament.

En primer lloc, s'han sospesat les debilitats que incorpora aquest projecte.

- Manca d'una direcció, és a dir, no hi ha cap pla d'evolució a llarg termini d'aquest projecte.
- El públic al qual s'enfoca és un públic restringit (desenvolupadors web), no és d'abast públic.

- Al ser un projecte estudiantil, es manca una financiació i es tenen recursos molt limitats.

En segon lloc, s'han definit una sèrie d'amenaces que afrontaria la empresa.

- Gestió pobre de les alertes, ja que només es tracta d'una sola persona i la resposta davant d'una alerta no sempre podria ser immediata.
- Existiria una competència amb pàgines que proporcionessin uns continguts similars.

Després, també s'han inclòs les fortaleses dins de la definició del projecte.

- S'ha considerat la incorporació d'exemples pràctics com a una fortalesa, al facilitar la comprensió de la teoria.
- La tecnologia utilitzada és Flask, de contingut dinàmic, això volent dir que es pot incorporar més material sense problemes.
- Es tracta d'una pàgina de contingut gratuït, sense restriccions basades en la economia de qui hi vulgui accedir.

Finalment, també s'han descrit una sèrie de punts per a les oportunitats futures.

- Possible introducció de nous apartats i categories, si es decideix ampliar l'inventari de vulnerabilitats explicades.
- Possibilitat de convertir-se una pàgina a la qual els desenvolupadors hi accedeixin habitualment per a consultar sobre vulnerabilitats.

3.2 Anàlisi inicial dels costos

S'ha fet un càlcul dels costos tenint en compte que es tracta d'un sol individual, l'únic objecte físic del qual s'ha fet ús ha sigut un ordinador, internet i s'ha hagut d'alquilar un servidor privat per a dur a terme el projecte. Per tant, s'han tingut en compte aquests aspectes per a fer el càlcul dels costos.

Si l'ordinador va costar una mitjana de 1200 euros, s'ha calculat que, si se li aplica un interès d'un 2%, al cap dels 11 mesos que s'ha trigat a fer el projecte:

$$2 * (1200 / 100) = 24 \text{ euros}$$

$$24 \text{ euros} * 11 \text{ mesos} = 264 \text{ euros d'amortització}$$

$$1200 + 264 = 1464 \text{ euros}$$

S'ha inclòs, també, el servidor dins de la categoria de consumibles, ja que el que es calcula per pagar cada més és la quantitat de recursos que utilitza la màquina virtual per a funcionar.

La màquina fa ús de al voltant de 10 euros al mes, els quals multiplicats per 11:

$$10 * 11 = 110 \text{ euros}$$

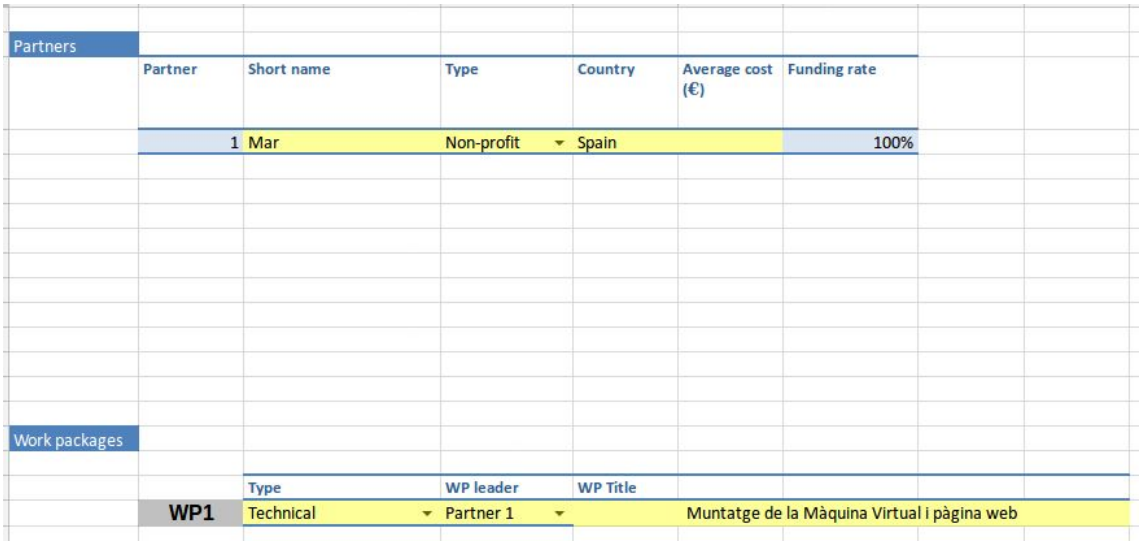
Finalment, s'ha calculat com a indirectes el cost del lloguer de l'habitatge , llum, internet i aigua, tot plegat sumant 375 euros totals.

		Partner	Short name	Type	Country	Average monthly cost (€)	Funding rate
		1	Mar	Owner	Spain		100%

		WP1	WP2	WP3	WP4	WP5	WP6	WP7	WP8	WP9	WP10	Total	
Cost categories	Personnel effort (time-units)											0	
	Personnel costs	0	0	0	0	0	0	0	0	0	0	0	0%
	Travel costs	0										0	0%
	Equipment (amortisation only)	1,464										1,464	75%
	Consumables	110										110	6%
	Other direct costs	0										0	0%
	Subcontracting	0										0	0%
	Total direct costs	1,574	0	0	0	0	0	0				1,574	81%
	Indirect costs	375	0	0	0	0	0	0				375	19%
	Total budget (eligible costs)	1,949	0	0	0	0	0	0	0	0	0	1,949	100%
Funding	**Requested EC funding**	1,949	0	0	0	0	0	0	0	0	0	1,949	

	Budget categories					Funding		
	Direct personnel costs	Other direct costs	Subcontracting costs	Indirect costs	Total eligible costs	Reimbursement rate	Maximum amount of the grant	Requested amount of the grant
Partner 1	0	1,574	0	375	1,949	100%	1,949	1,949

	Budget categories					Funding		
	Direct personnel costs	Other direct costs	Subcontracting costs	Indirect costs	Total eligible costs	Reimbursement rate	Maximum amount of the grant	Requested amount of the grant
Partner 1	0	1,574	0	375	1,949	100%	1,949	1,949
Total consortium	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!



Partners

Partner	Short name	Type	Country	Average cost (€)	Funding rate
1	Mar	Non-profit	Spain		100%

Work packages

	Type	WP leader	WP Title	
WP1	Technical	Partner 1		Muntatge de la Màquina Virtual i pàgina web

Fins al moment no s'han trobat altres paquets de treball que s'hagin de posar de forma addicional, però estan subjectes de ser modificats al llarg del projecte, ja que aquesta només és la estimació inicial.

3. 3 Eines per a la gestió

Per a la gestió del projecte, s'han utilitzat una sèrie d'eines extretes de fonts online i software oficial:

- Diagrames de Gantt: per a realitzar el diagrama de Gantt, s'han utilitzat els recursos que Google Drive posa a la disposició dels seus usuaris. Es va crear un document Excel online, el qual es guardava en format xlsx cada vegada que es modificava.
- Anàlisis DAFO: per a realitzar el anàlisis DAFO s'han utilitzat els recursos oferts per la pàgina www.canva.com, la qual ofereix múltiples plantilles per a la creació de DAFO.

4. Metodologia

Descripció pas a pas, les diferents fases i subfases del projecte.

4.1 Pluja de idees

Es va escriure tot el que es tenia en ment quan es pensava en els objectius futurs del projecte.

Per a dur a terme aquesta fase, va ser necessari fer una primera recopilació d'informació per a determinar quin tipus de tecnologies i informacions hi havia disponibles.

A l'hora de fer aquest primer recull d'informació, es van tenir en compte una sèrie d'aspectes que es consideraven importants:

- ➢ Facilitat d'implementació: havien de ser eines senzilles de implementar, les quals no provoquessin problemes de compatibilitat amb els altres programes.
- ➢ Amb una bona documentació: es cas de que sorgissin problemes, la recerca d'una solució havia de resultar fàcil i ràpida de trobar.
- ➢ Possibilitat de seguir el projecte en cas de que es volgués continuar.

Una vegada considerats tots aquests aspectes, es van anotar totes les possibilitats restants.

Aquestes opcions es van anar descartant a mesura que, o bé es trobaven aspectes que no acabaven d'encaixar del tot, o bé en tests pràctics d'implementació no resultaven en el comportament que s'esperava.

Per exemple, dos bons exemples en quant a tecnologia i contingut són AngularJS i una categoria anomenada *Key-Value Databases.*

AngularJS

Al quedar aquest *framework* en JavaScript com una de les opcions, es va fer una comprovació inicial.

Durant la comprovació, es va fer obvi que la implementació no era senzilla, ja que requeria una gran quantitat de dependències i molt d'espai, el que resultaria en un gran volum de recursos utilitzats en el servidor i molt de temps emprat a l'hora de fer recerca i solució d'errors. Per tant, es va descartar.

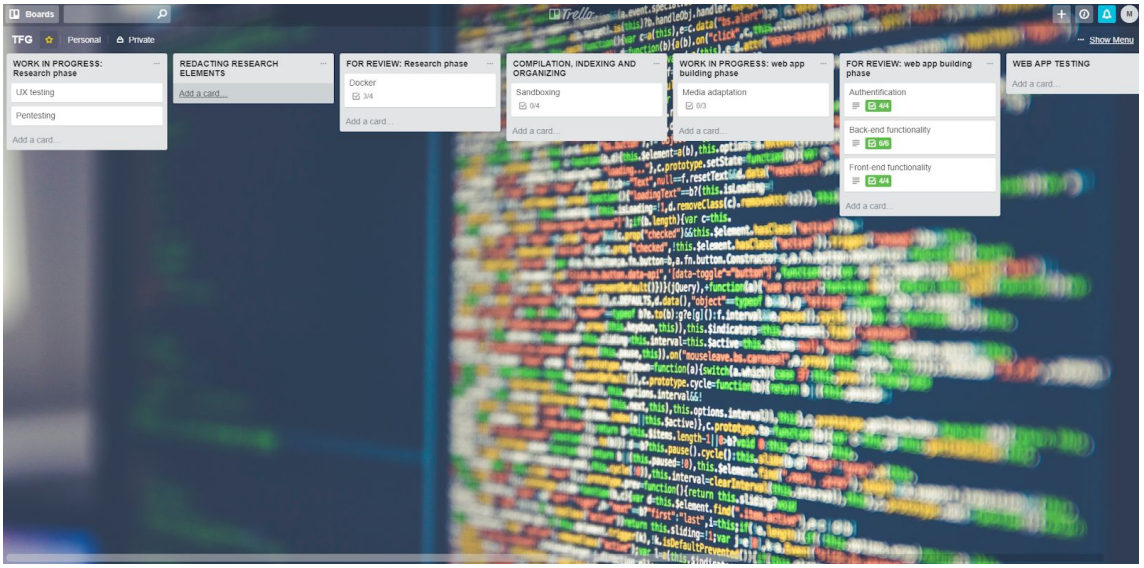
Key-Value Databases

Inicialment es va incloure aquest subapartat dins de l'apartat explicant les vulnerabilitats pertanyents a les bases de dades.

Durant la revisió de contingut, es va veure que no resultava pràctic incloure-ho, ja que els seus continguts resultaven ser específics per a cada base de dades concretes i resultava difícil, si més no impossible, generalitzar-ho.

4.2 Creació d'un taulell Kanvan amb Trello

Es va procedir a la creació de diversos taulells mitjançant l'eina digital Trello, amb la qual es van fixar diferents passos per els que s'havia de passar abans de donar un projecte per finalitzat.



Es van crear una sèrie de taules específiques per cada fase del projecte i l'estat de les diferents fases.

Tot i que alguns taulells estiguin marcats com a completats, no s'han desat en un taulell de "completat" perquè encara cal fer les últimes revisions i comprovacions abans de donar el projecte per finalitzat.

4.3 Implementació de la teoria sobre la pràctica

Es van incorporar totes les dades teòriques generades sobre la tecnologia amb la qual s'anava a treballar.

Per a fer-ho, es va organitzar el temps de recerca i redacció, amb una mitjana d'una hora i mitja al dia per a tal. A continuació, un cop totes les

dades van ser recopilades, es va tornar a aplicar la mateixa tècnica i temps diari per a la seva redacció dintre de la pàgina web.

Els diagrames de Gannt van resultar molt útils per a veure el desenvolupament del projecte a llarg plaç, i per poder fer una estimació del temps que s'hauria d'aplicar dins del projecte per a tenir-lo completat en un marge de temps òptim.

4.4 Test de funcionalitat

Per tal de comprovar que tot el que es va desenvolupar funciona com s'espera, es va implementar una part de test del treball.

➔ La primera subfase consisteix en un test enfocat a l'UX, és a dir, per a comprovar si el disseny centrat en l'usuari ha sigut implementat correctament, si existeixen punts millorables o d'altres que s'haurien de suprimit o modificar, si seria necessari modificar els esquemes de colors o la disposició dels elements, etc...

➔ La segona subfase consistirà en un test enfocat a provar la seguretat de la pàgina web. En primer lloc, es faran varies proves sobre la pàgina web dutes a terme per persones no externes. En segon lloc, es convidarà a persones externes a tractar de trobar forats en la seguretat de la pàgina web, per tal d'arreglar qualsevol error en la seguretat de la pàgina web.

5. Desenvolupament web

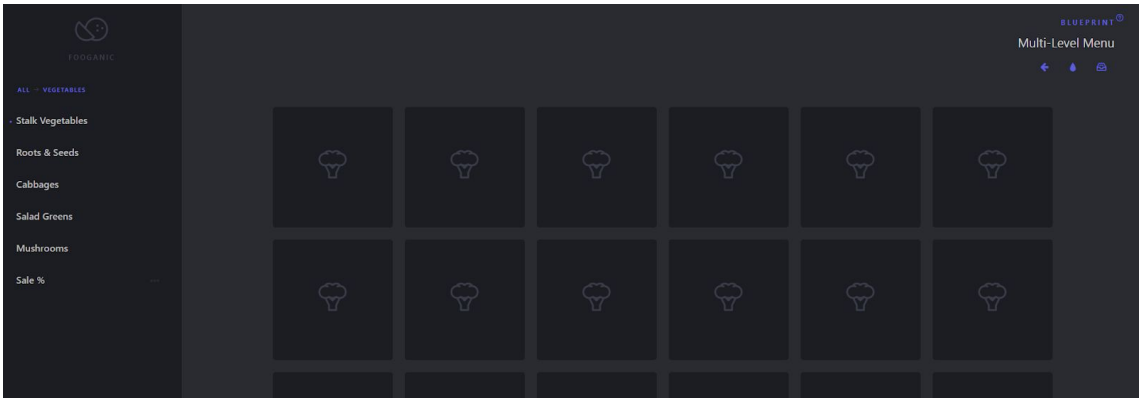
5.1 Disseny de la pàgina web

El primer que es va creure necessari, abans de començar amb el desenvolupament web, va ser escollir l'estètica que hauria de tenir la pàgina web.

Es volia que fos una estètica minimalista i futurista, alliberant-la de botons innecessaris i de qualsevol distracció dels objectius principals.

Per això, es va escollir una plantilla senzilla i sense complicació, on l'únic contingut fos també el contingut principal de la pàgina.

Amb l'ajut de la web www.tympanus.net, es va trobar una plantilla d'ús public que responia a les qualitats que es buscaven.



Com es pot comprovar, la pàgina constava de un menú a l'esquerra mostrant totes les categories a escollir. Les categories es mostren en una sèrie de requadres grisos amb separació uniforme.

El fet de que el menú a la columna de l'esquerra mostrés les lletres en blanc permet que l'usuari vegi en tot moment clarament en quin nivell es troba dins de la navegació de la pàgina web. D'altra banda, els requadres grisos li donen un ordre i mantenen neutre l'estètica de la pàgina.

5.2 Què és Flask?

Abans d'aprofundir sobre com es va fer ús d'aquesta tecnologia per tal de crear la pàgina web, es farà una descripció sobre com funciona i de quina manera es pot jugar amb la seva lògica per a mostrar continguts web.

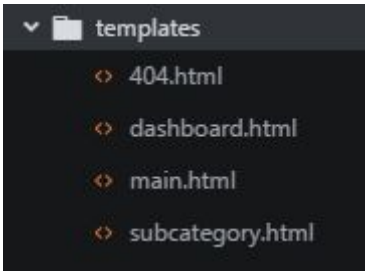
Segons la pàgina web oficial de Flask[7], es tracta d'una *microframework*, és a dir, d'una plantilla de treball per als desenvolupadors web amb els

recursos únicament necessaris. Flask oficialment només ofereix tot allò que és absolutament necessari per la seva creació i implementació, tot i així, usuaris arreu de la web creen cada dia llibreries per donar noves funcionalitats a Flask, ja que es tracta de un projecte *open source,* en el qual qualsevol usuari pot participar en el seu creixement[8].

S'explicarà el funcionament de Flask amb exemples extrets d'aquest mateix projecte.

Primerament, el desenvolupador haurà de crear tantes plantilles com pàgines mostri la web, és a dir, una plantilla per a cada secció. Per exemple, a l'hora de crear les seccions per aquest projecte es van crear 4 plantilles diferents:

- 404.html: pàgina genèrica d'error per quan un usuari navegui a una pàgina inexistent.
- dashboard.html: pàgina inicial a la que el usuari arribarà per primera vegada.
- main.html: pàgina que mostrarà les caselles amb les diferents categories.
- subcategory.html: pàgina per a cadascuna de les diferents categories.



Dins de cada plantilla, existeix una lògica per a mostrar el contingut web, la qual s'ha d'escriure amb el llenguatge de programació Python.

Per exemple, si es volgués crear quatre *tags* HTML amb diferents números al *id*, es faria d'aquesta forma:

```
{% for string in element %}
<form action="subcategory.html" method="POST">
  <li><button class="items" href="subcategory.html" name="{{id}}" type="submit" value="submit">
    <h5>{{string}}</h5>
  </button></li>
</form>
{% endfor %}
```

Amb el *for loop* farem que es creïn tants tags HTML com elements hi hagi en *element*. Però, com se sap quantes vegades ha d'entrar dins el *loop*?

Per extreure el valor que tindrà el valor de totes les paraules clau que apareixen entre cortxets ({{}}), així com les variables que apareixen en els condicionals (for, if, while, ...), existeix un altre document a part purament escrit en Python (els mostrats fins ara barregen Python amb HTML) que passa tots aquests valors a mesura que es carreguen els recursos web.

Aquest document rep el nom de __init__.py, i és el document que Flask buscarà sempre perquè pugui funcionar com a tal.

__init__.py retorna les plantilles web amb tota la informació necessària.

```
if category == 'Authentication' or 'Front-end' or 'Back-end':
    if subcategory == '':
        return render_template("dashboard.html", category=pretty(categories_data), titles=titles_data)
```

El *render_template* diu a Flask quina és la plantilla a mostrar (en aquest cas dashboard.html), i a continuació les variables que contindrà aquesta plantilla. *Category* i *titles* contenen informació extreta de la base de dades (s'ha assignat aquest nom a la informació més a dalt al document i no es mostra en la captura), i per tant, quan dashboard.html llegeixi les paraules *category* i/o *titles*, sabrà que es refereix a la informació que hi ha dins de la base de dades.

```
<h1> This is the {{ category }} part </h1>
```

This is the Authentication part

Com es pot apreciar, la plantilla assigna el significat que ha rebut del fitxer central, __init__.py, amb la informació que s'ha passat en forma d'arguments. En aquest cas, el que ha rebut ha sigut la paraula en clau *category*, i ha assignat aquesta paraula que venia entre cortxets amb la paraula extreta de la base de dades *Authentication.*

5.3 Perquè es va escollir Flask?

Els motius pels quals es va escollir treballar amb aquesta tecnologia són diversos i es presentaran en una sèrie de punts:

- Familiaritat: degut a que ja es va treballar en el passat amb aquesta *framework* per la creació d'altres projectes, ja es tenia una certa familiaritat amb el seu funcionament. No només això, sinó que també, al tenir coneixements de programació amb Python, no era necessari aprofundir en la comprensió de les lògiques que feien funcionar tot.
- Senzillesa: tot i que la col·laboració de centenars d'usuaris han fet que Flask compti amb diverses llibreries per usos molt diferents, la estructura bàsica i el seu funcionament són molt senzills, provocant això que, davant de l'aparició d'errors o *bugs* en el sistema, trobar aquests errors també resulta senzill.
- Possibilitat de compaginar amb altres tecnologies: especialment amb les diferents bases de dades que hi ha, cada desenvolupador empra la que més bé s'adapta a les seves circumstàncies. Degut a aquest fet, ja se sabia que qualsevol base de dades que s'escollís seria perfectament adaptable amb Flask.
- Contingut dinàmic: de cara al futur, si es decidís continuar amb el creixement d'aquesta pàgina web, el fet de que tingui contingut dinàmic fa que la incorporació de nou material sigui senzill i que no neixi la necessitat de reestructurar la lògica interna ni el codi HTML.
- Popularitat: degut a que és una Framework tant popular, qualsevol error es pot trobar dins els forums o fils de preguntes i respostes entre els mantenidors i els usuaris.

5.4 Incorporació de Flask amb el disseny escollit

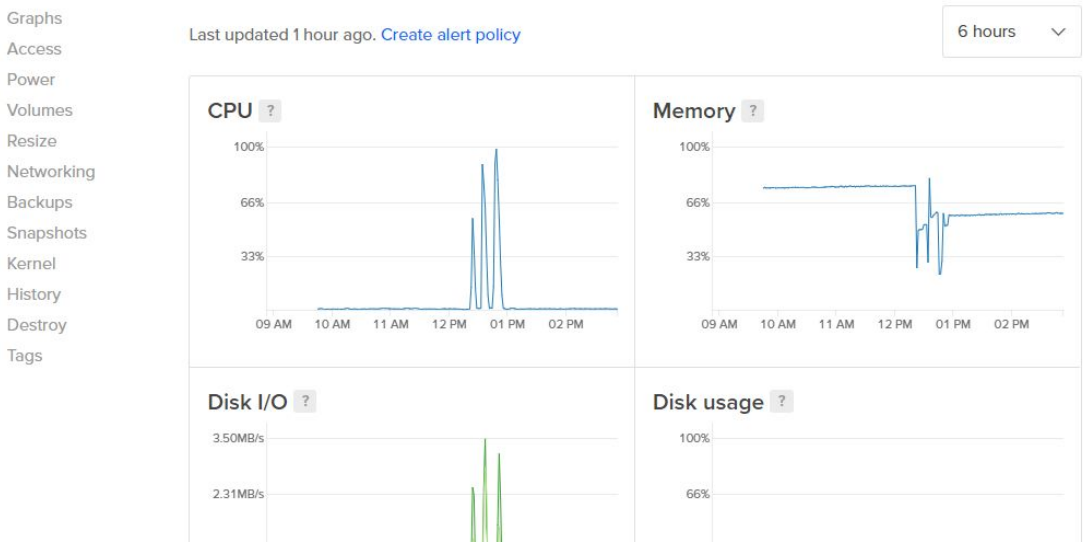
Per incorporar Flask dins del disseny escollit, van haver de realitzar-se modificacions dins les plantilles descarregades. Aquestes modificacions van consistir en diversos passos, i per explicar-los en detall, es procedirà a explicar com es va instal·lar Flask, seguit de la comparació del disseny extret de la web i el disseny i desenvolupament final, per a tenir un punt de comparació.

5.4.1 Preparant Flask

Per instal·lar Flask correctament, primer era necessari fer ús d'un servidor privat virtual (VPS). Dins d'aquest VPS es podien obrir ports a internet perquè la pàgina fos visible i visitable.

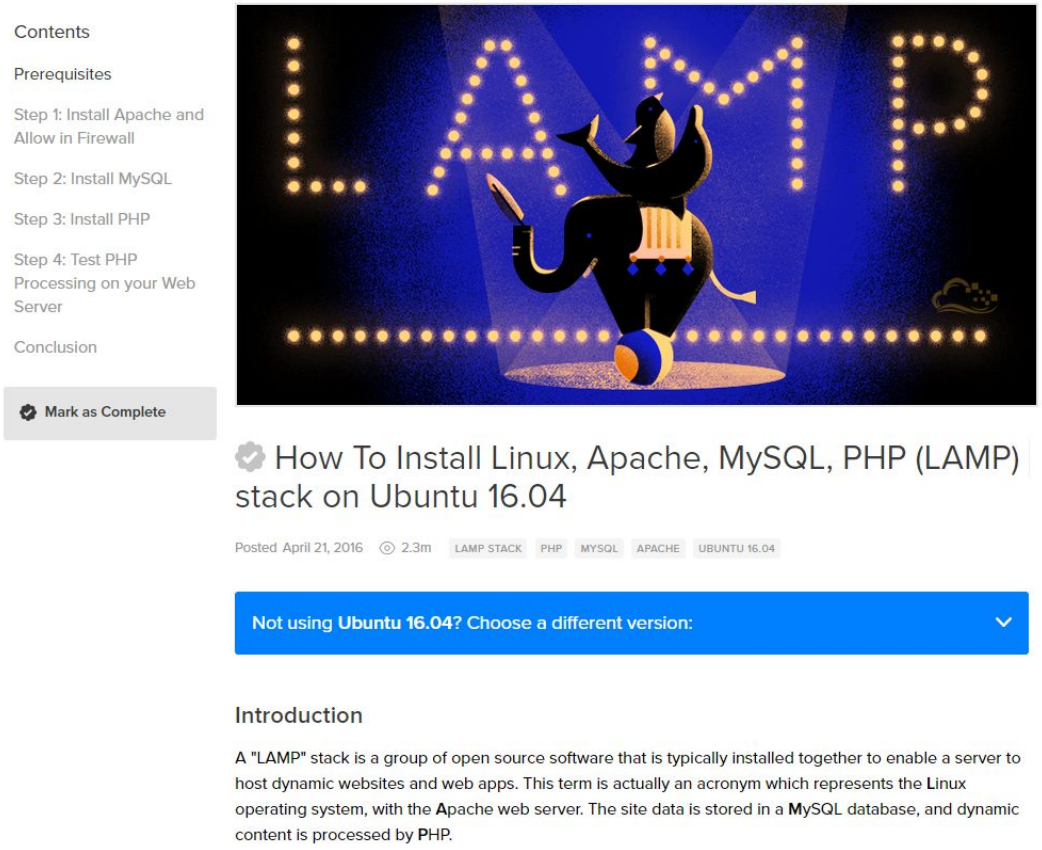
Es va escollit DigitalOcean[9] com a plataforma per a emmagatzemar les dades y obrir el servidor al núvol. El motiu va ser, no només els baixos

preus, sinó també totes les possibilitats que ofereix per combinar amb el VPS.

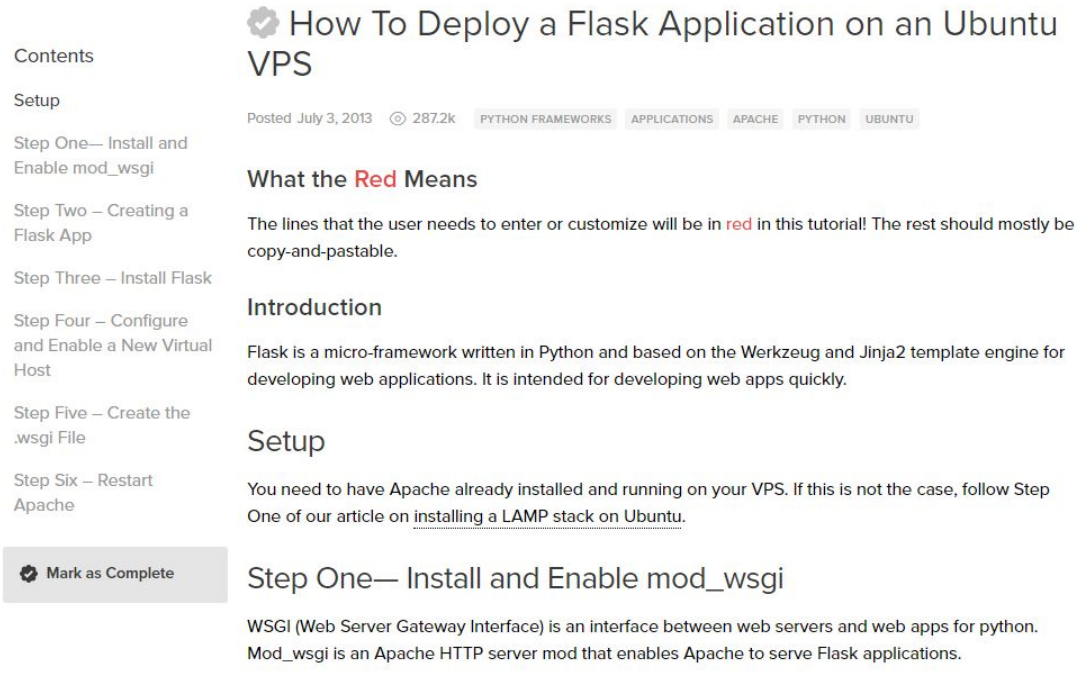


Abans d'obrir els ports, era necessari tenir un servidor des del qual es poguessin servir aquests continguts. Es va escollir un servidor Apache2 per la seva popularitat i facilitat de integració amb MySQL.

DigitalOcean conté molta informació sobre com realitzar diverses accions dins dels seus VPS, gràcies a la qual es va poder trobar una guia sobre com instal·lar Apache2 i MySQL[10] dins d'un dels seus servidors.



Un cop Apache2 i MySQL estaven correctament instal·lats i configurats, va ser el torn d'instal·lar tot el que corresponia a Flask. DigitalOcean també té una guia per la seva instal·lació i configuració[11].



Una vegada instal·lat, donat que Flask requereix un ambient virtual, és a dir, un directori completament aïllat de tot el sistema de carpetes del sistema operatiu, es va haver de tornar a instal·lar Apache2 i MySQL dintre de l'ambient virtual fent ús de la primera guia.

Un cop es va acabar, ja es podia començar a adaptar la plantilla per al seu us amb Flask.

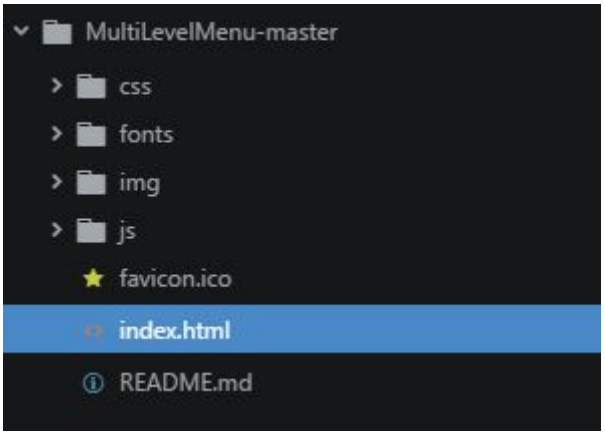
Durant aquesta fase no van haver problemes de cap tipus, ja que les accions eren unidireccionals i la documentació que DigitalOcean ofereix està molt completa davant de totes les direccions que pugui prendre una instal·lació.

5.4.2 Modificació dels elements HTML per a ser utilitzats amb Flask

Primerament, era necessari crear un fitxer anomenat __init__.py, el qual Flask reconeix i empra perquè totes les plantilles que formen part de la mateixa web es puguin reconèixer entre sí, a més de les dades de les bases de dades.

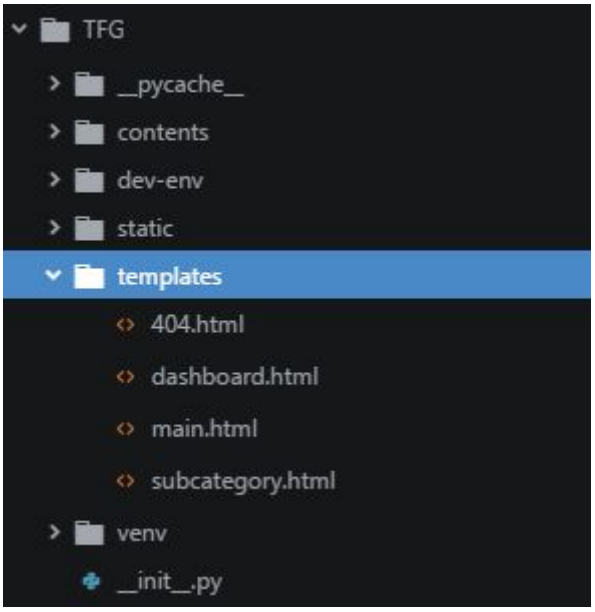
En aquesta secció es parlarà únicament de l'HTML que s'ha hagut de canviar i modificar i de la lògica que s'ha creat tant a __init__.py com als fitxers HTML individuals.

En primer lloc, es va observar que la plantilla web original només contenia un sol document HTML, el qual feia servir per mostrar tota l'estructura de la web.



Aquest fet ocasionava que la idea inicial (separar les diferents pàgines en diverses plantilles HTML) es trenqués.

Amb lo qual, es va haver de separar tot aquest contingut en diferents fitxers.



El que es va fer en aquest pas va ser dividir el que existia en el primer document HTML en diversos documents HTML.

Seguidament, es va començar a escriure el que contindria __init__.py

```python
from flask import Flask, render_template, request
from flask_htpasswd import HtPasswdAuth
import string

from dbconnect import connection

app = Flask(__name__)


@app.route('/')
@app.route('/dashboard/')
def homepage():
    return render_template("main.html")
```

Es van importar totes les llibreries necessàries i es va reconduir tot el tràfic que anés als directoris principals cap al document main.html.

El document main.html conté tots els estils CSS i JS, els quals serien carregats en els altres documents gràcies a que la tecnologia Flask permet incorporar documents HTML dins d'altres documents HTML.

```
{% extends "main.html" %}

{% block body %}

<h1> This is the {{ category }} part </h1>

<ul>
  <li class="category-title content">
    <ul class="category-items content">
    {% for element in titles %}
    {% set id = loop.index %}
      {% for string in element %}
      <form action="subcategory.html" method="POST">
        <li><button class="items" href="subcategory.html" name="{{id}}" type="submit" value="submit">
          <h5>{{string}}</h5>
        </button></li>
      </form>
      {% endfor %}
    {% endfor %}
    </ul>
  </li>
</ul>

{% endblock %}
```

En la part superior, es pot veure que {% extends (...) %} inclourà el document HTML especificat dins d'aquella mateixa plantilla.

Finalment, es van introduir les paraules clau o variables dintre de cada plantilla.

```
{% extends "main.html" %}

{% block body %}

<h1> This is the {{ titles[name][0] }} part </h1>

{{subcategory[name][0] | safe}}

<div class="button-block">
  <a href="#" class="btn button-lab">Click me to perform a successful attack!</a>
</div>

{% endblock %}
```

Aquestes paraules clau obtenien el seu significat a partir de les dades que eren traspassades des de __init__.py, que al seu torn obria una connexió amb la base de dades i extreia els continguts.

```python
@app.route('/dashboard/<dir_category>/')
@app.route('/dashboard/<dir_category>/<subcategory>', methods=['POST', 'GET'])
def dashboard(dir_category, subcategory=None):
    c, conn = connection()

    categories_query = "SELECT [redacted] FROM [redacted] WHERE [redacted] = '%s'" % (dir_category)
    c.execute(categories_query)
    categories_data = c.fetchall()
    subcategories_query = "SELECT [redacted] FROM [redacted] WHERE [redacted] IN (SELECT [redacted] FROM [redacted] WHERE [redacted] IN (SELECT [redacted] FROM [redacted] WHERE [redacted] = '%s'))
    c.execute(subcategories_query)
    subcategories_data = c.fetchall()
    titles_query = "SELECT [redacted] FROM [redacted] WHERE [redacted] IN (SELECT [redacted] FROM [redacted] WHERE [redacted] IN (SELECT [redacted] FROM [redacted] WHERE [redacted] = '%s'))" % (dir_
    c.execute(titles_query)
    titles_data = c.fetchall()
```

Era necessari, a més, un altre fitxer en Python que gestionés la connexió amb la base de dades, el qual es va anomenar dbconnect.py

Les dades que eren extretes de la base de dades d'aquesta forma, es passaven a la lògica d'enrutament de Flask.

```python
if category == 'Authentication' or 'Front-end' or 'Back-end':
    if subcategory == '':
        return render_template("dashboard.html", category=pretty(categories_data), titles=titles_data)
    else:
        if request.method == 'POST':
            result = request.form
            r = ''
            for r in result:
                r = int(r) - 1
            return render_template("subcategory.html", subcategory=subcategories_data, titles=titles_data, name=r)
else:
    return render_template("404.html")
```

Algunes de les imatges han sigut modificades amb rectangles negres per protegir la informació privada.

5.5 Introducció del contingut dins la web

Es va observar que la plantilla original incloïa les dades que presentava a la web fent ús de la tecnologia Ajax.

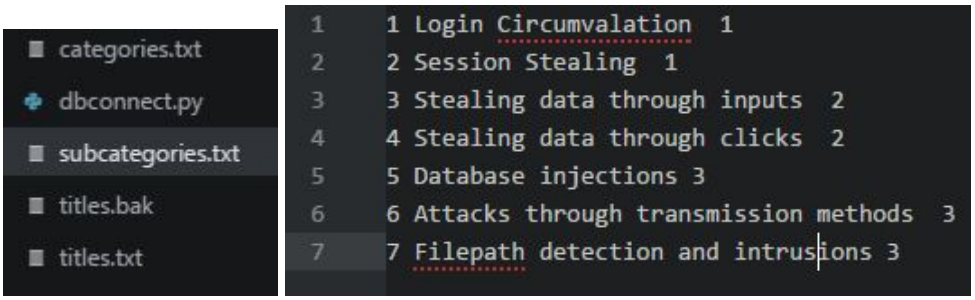
```html
<div class="content">
  <p class="info">Please choose a category</p>
  <!-- Ajax loaded content here -->
</div>
```

```
// grid dummy data
var dummyData = {
  "Stalk Vegetables":"<li class=\"pr
  class=\"product\"><div class=\"foo
  foodicon--broccoli\"></div></li><l
  class=\"product\"><div class=\"foo
  "Roots &amp;amp; Seeds":"<li class=\"p
  class=\"foodicon foodicon--grain\"
```

D’altra banda, el projecte propi necessitava obtenir la informació de dins de una base de dades.

El que es va fer va ser, de totes les dades recopilades en un document Word, es va procedir a fer un resum, no només dels continguts, sinó també de les categories i subcategories.

Aquest resum es va passar a un fitxer txt.



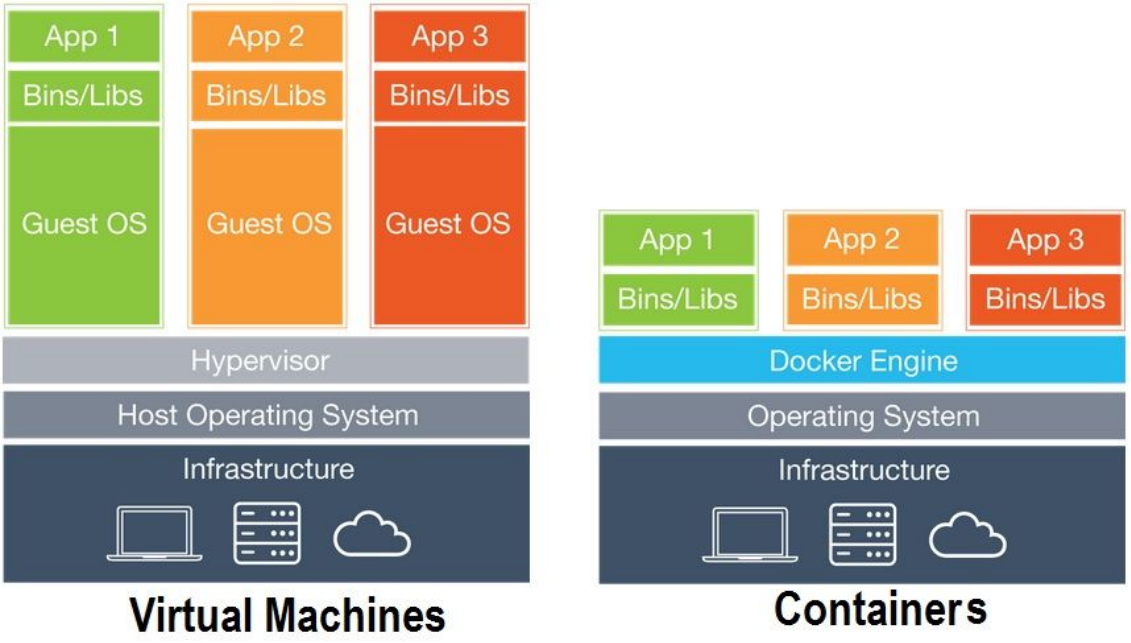
A continuació es va accedir a través de FileZilla al directori local i es van pujar els documents TXT. Finalment, aquests documents TXT es van incorporar dins la base de dades fent servir les *queries* de MySQL per a la gestió de les dades.

6. Creació d'un contenidor aïllat

6.1 Què és Docker?

Per poder començar a explicar com i perquè es va utilitzar Docker, és necessari explicar què és exactament i perquè està guanyant tanta popularitat.

Docker és una tecnologia sorgida a l'any 2013, nascuda de la necessitat de virtualitzar ambients de treball sense la necessitat de crear màquines virtuals per aquesta tasca. Els creadors de Docker pretenien allunyar-se de la funcionalitat de màquines virtuals, ja que aquestes requerien la incorporació del kernel sencer, mentre que Docker fa servir el kernel de la màquina inquilina.



Il·lustració 11 - Comparació d'una màquina virtual amb una màquina de Docker

Cada instància de Docker rep el nom de contenidor, i cada contenidor es forma a partir d'una imatge.

Les imatges de Docker contenen tota la informació vital que compondrà el sistema operatiu i poden funcionar sense ser modificades, però necessiten un contenidor per prestar els seus serveis. Diversos contenidors poden fer servir la mateixa imatge però diferents imatges no poden fer servir un mateix contenidor.

Aquesta tecnologia, a més, facilita la modificació de imatges i contenidors d'una manera més senzilla i menys exhaustiva per als recursos de la

màquina inquilina, de forma que aplicar modificacions, parar i tornar a encendre el contenidor no comportarien tant treball per a la CPU com ho faria una màquina virtual.

Per últim, Docker incorpora recursos integrats de seguretat, motiu que va fer que aquest projecte es decantés pel seu ús.

6.2 Perquè es va escollir Docker?

Els motius pels quals es va escollir Docker poden ser resumits en una sèrie de punts:

- Facilitat d'ús: Docker només necessita una sèrie de comandes dins del sistema operatiu emprat per a instal·lar-se i començar a ser utilitzat. A més a més, la pàgina oficial ofereix una extensiva documentació per a les persones novícies trobin senzill el seu ús i res quedi sense comprendre.
- Comunitat: els creadors de Docker mantenen no un, sinó diversos projectes *open-source*, als quals la comunitat hi pot contribuir, aportant millores o corregint errors. Aquest fet marca una diferència molt important, ja que significa que els errors trigaran menys a ser trobats i corregits i les millores comptaran amb els suport del públic.
- Seguretat integrada: aquesta era una propietat que va ser un factor important a l'hora de decidir utilitzar Docker, ja que el projecte en sí consisteix en seguretat web, i es volia aconseguir que tots els recursos que han sigut creats o carregats dins aquest projecte fossin el més segurs possibles.
- Recursos utilitzats: els recursos que són utilitzats per cada contenidor són mínims comparats amb el que consumiria apagar i reiniciar una màquina virtual sencera. També era necessari estalviar recursos, ja que DigitalOcean marca uns límits els quals el servei alquilat no pot sobrepassar cada més. Tot i així, en un moment va haver problemes, els quals seran explicats més endavant.

6.3 Passos necessaris per a la creació del contenidor

De la mateixa forma que amb Flask, es va seguir la documentació oficial[12] de la pàgina de Docker per a començar a fer funcionar el projecte.

En primer lloc, DigitalOcean també inclou documentació sobre com instal·lar Docker en un dels seus VPS[13], per tant es va seguir la documentació fins que es va tenir tot el necessari per arrencar i fer servir Docker dins la màquina.

A continuació, no es va seguir la part 2 de la documentació de Docker, ja que explicaven com començar una aplicació de Flask dins del contenidor. Tot i que s'hagués utilitzat Flask fora de Docker, no es volia utilitzar dins, sinó que es pretenia utilitzar un simple servidor Apache2 dins de un contenidor amb Ubuntu.

Per tant, es va seguir un altre guia oferta per la documentació oficial de Docker[14]. En aquest documentació es descrivia com poder començar una imatge de Docker personalitzada.

Primerament era necessari descarregar-se la imatge base sobre la qual es basaria la imatge personalitzada. Es va escollir una imatge de Ubuntu 16.04 per facilitat d'ús i familiaritat. A continuació es va utilitzar aquesta imatge base per crear una nova imatge anomenada web_test, dins la qual es va instal·lar Apache2 i res més, ja que la imatge base tenia tot el necessari pel seu funcionament.

```
root@TFG:/var/www/FlaskApp/FlaskApp/dockertest# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
web_test            0.1                 a04419724fea        2 months ago        282MB
ubuntu              16.04               0458a4468cbc        2 months ago        112MB
```

Per comunicar a la nova imatge que s'havia de basar en la imatge base de Ubuntu i instal·lar Apache2, era necessari crear un document el qual, per configuració pròpia, la imatge buscaria quan fos llençada. Aquest document rep el nom de Dockerfile.

```
FROM ubuntu:16.04
ADD . /code
WORKDIR /code
EXPOSE 80
RUN apt-get -y update && apt-get -y upgrade
RUN apt-get install -y apache2
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Com es pot comprovar, se li comunica que s'ha de basar en Ubuntu i instal·lar Apache2.

Després de tenir la imatge llesta, era el torn de començar el contenidor basat en la imatge creada.

Abans de començar-lo, es va pensar en què es necessitava per aquest contenidor. Per explicar-ho bé, primer es compartirà la imatge de la *query* per al funcionament del contenidor i s'explicarà després perquè cadascun dels paràmetres eren necessaris.

docker run -it --security-opt seccomp=policy.json --name web_labs -d -p 1337:80 -v /var/www/html:/var/www/html a04419724fea

Es saltarà la secció de seccomp, ja que forma part del següent punt.

- ➔ --name web_labs: determina el nom que rebrà aquest contenidor
- ➔ -d -p 1337:80 el port que s'obrirà cap a internet. El contenidor obrirà el seu port 80, el qual serà accessible des d'internet a través del port 1337.
- ➔ -v /var/www/html:/var/www/html: són les carpetes compartides. Per pujar el codi HTML dins del contenidor, ja que no s'hi pot accedir a través de FileZilla (és un servei virtualitzat), era necessari crear carpetes compartides dins de la màquina inquilina.
- ➔ a04419724fea: el identificador de la imatge de web_test, perquè el contenidor l'utilitzi com a imatge base.

Amb tots aquests passos, ja es tenia el contenidor de Docker funcionant amb tot el necessari perquè prestés un servei web.

Un dels últims passos en quant a contenidor era aïllar-lo correctament, per tal d'impedir que un intrús no desitjat es pogués escapar de dins del contenidor i tenir accés a les dades de la màquina inquilina, així com de la seva base de dades.

6.4 Mecanismes per aïllar el contenidor

Per completar aquesta tasca, era necessari definir què es volia i trobar la funcionalitat necessària per a tal. Preferiblement, es buscaven comandes que foren natives del sistema operatiu Linux per a limitar les descàrregues.

Es mostraran les comandes que es van escollir i el perquè.

- chroot: chroot permet canviar el directori inicial des del qual es comença un projecte.
- seccomp: seccomp limita els *syscalls* que el sistema operatiu pot fer.
- setuid/setgid: setuid i setgid dona o retira permissos d'usuaris i de grups d'usuaris

- apparmor: apparmor restringeix l'activitat de xarxa que hi pot haver dins del sistema operatiu, així com també l'execució de programes.

6.4.1 Chroot

Es va considerar necessari aplicar chroot amb la intenció de que, si un atacant aconseguís entrar dins del contenidor de Docker, no es trobés en el directori principal de Linux, des d'on podria observar tota la estructura interna i extreure informació, encara que no pogués executar res.

Amb chroot s'aconsegueix que el directori d'entrada al sistema sigui un directori creat especialment amb aquesta finalitat.

Docker ofereix la possibilitat de crear un directori per aquesta finalitat alhora que es crea el contenidor. La forma de fer-ho es especificant-ho en *Dockerfile.*

```
FROM ubuntu:16.04
ADD . /code
WORKDIR /code
EXPOSE 80
RUN apt-get -y update && apt-get -y upgrade
RUN apt-get install -y apache2
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Les línies *ADD . /code* i *WORKDIR /code* creen un directori al *root* del sistema i fan que el directori de treball sigui aquest. D'aquesta forma, quan l'administrador o un atacant entri per primera vegada dins del contenidor, estarà dintre del directori */code* i no *root*.

```
root@TFG:/var/www/FlaskApp/FlaskApp/dockertest# docker exec -it web_labs /bin/bash
root@ab2718088f90:/code#
root@ab2718088f90:/code#
root@ab2718088f90:/code#
root@ab2718088f90:/code#
root@ab2718088f90:/code#
root@ab2718088f90:/code# ls
Dockerfile  requirements.txt
root@ab2718088f90:/code#
```

Com s'observa en aquesta imatge, al entrar dins del contenidor *web_labs*, s'entra directament a */code*.

6.4.2 Seccomp

Es va aplicar seccomp amb la intenció de que algunes crides de sistema foren il·legals. Aquestes crides del sistema es van il·legalitzar ja que

permetien dur a terme accions que podrien causar que un atacant es comuniqués amb altres zones d'internet o inclús sortís del contenidor.

La manera d'aplicar seccomp dins de un contenidor Docker és especificant-ho quan es llença el contenidor, ja que Docker ofereix la opció de fer-ho.

docker run -it --security-opt seccomp=policy.json --name web_labs -d -p 1337:80 -v /var/www/html:/var/www/html a04419724fea

La opció per incorporar un fitxer amb les *syscalls* permesses dins del sistema es fa amb la opció *--security-opt seccomp=policy.json*

S'ha aplicat per algunes *syscalls*, però no s'ha aconseguit del tot el que es pretenia, ja que existeixen *syscalls* que són necessàries per la creació del sistema operatiu, i per tant no es poden eliminar.

6.4.3 Setuid/Setgid

Aquestes comandes es van escollir amb la intenció de crear un usuari sense privilegis, i d'aquesta forma fer que qualsevol atacant no tingués privilegis si entrés dins el contenidor.

No s'ha aconseguit encara pel fet de que, quan es crea el contenidor de Docker, és necessari que es faci tot desde *root*, donat que es realitzen tasques que requereixen el nivell més privilegiat.

El problema es troba en que no s'ha aconseguit trobar la forma de crear l'usuari sense privilegis i fer que sigui l'usuari per defecte després de la creació del contenidor.

6.4.4 AppArmor

Amb AppArmor es pretén impedir que hi hagi connexions cap a l'exterior, ja que hi ha *syscalls* que són imprescindibles i no es poden excepcionar amb seccomp (el sistema operatiu necessita descarregar-se Apache2 segons se li ha indicat quan està creant el contenidor).

Encara no hi ha hagut temps d'aplicar aquesta comanda, pel qual es farà durant la tercera part del treball de final de grau.

6.5 Disseny del contingut web del contenidor

Per últim, es parlarà del contingut web que s'ha pujat dins del contenidor i que es troba actualment obert a internet.

6.5.1 Intenció de la web

La intenció principal de la web creada dins del contenidor era fer que, aquells usuaris que haguessin llegit sobre les vulnerabilitats en la web principal, poguessin veure una mica més de la part pràctica i sobre com es veuria o es duria a terme en un ambient més realistic.

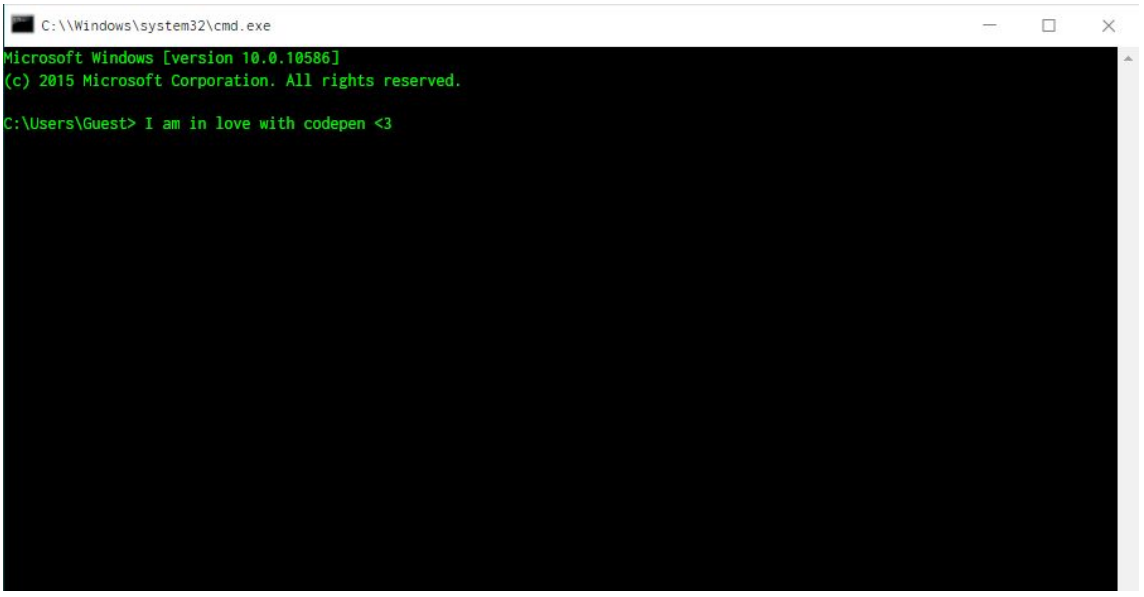
Es van simular línies de comandes o ús d'eines reals per a que semblés més proper i costés menys d'imaginar.

A continuació s'exposarà com es va dur a terme.

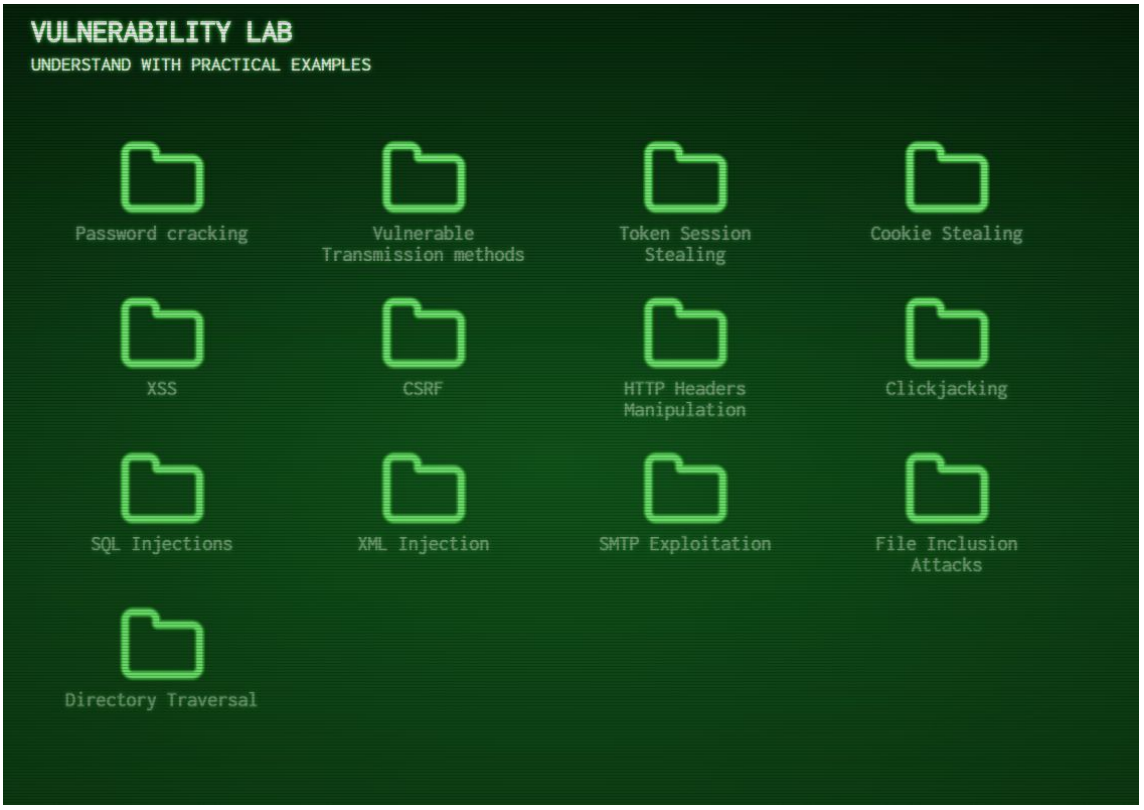
6.5.2 Disseny de la web

Per el disseny de la pàgina web es va preferir un disseny el qual oferís un ambient diferent. Es va decantar per un disseny el qual mantenís a l'usuari curiós sobre els continguts de la pàgina web, de la mateixa forma que perdés la serietat de les pàgines anteriors.

Finalment, després de un temps de recerca, va sorgir la idea de emular una terminal d'ordinador. Es va prendre com a base els colors i tipografies emprats en la creació pública *Windows shell*[15], per Fabien Gréard.



També se li va donar un fons de terminal antiga i es van associar les diferents seccions en les quals es repartien les diferents vulnerabilitats amb imatges de carpetes.



Els resultats són els que es mostren a la imatge.

Cadascuna de les seccions conté la part pràctica de la secció que li pertoca. Tot i això, no està complert del tot, ja que falta omplir informació, tasca la qual es farà en la tercera etapa del treball de fi de grau.

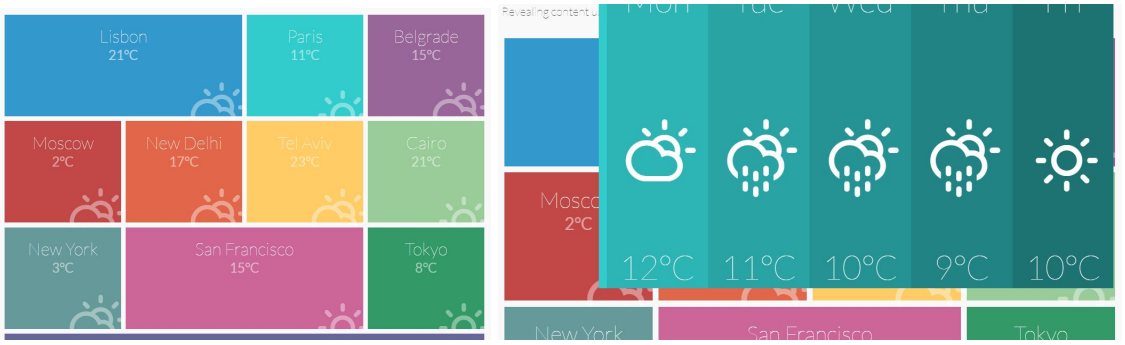
A continuació s'explicarà com es va construir la plantilla HTML, amb els format CSS.

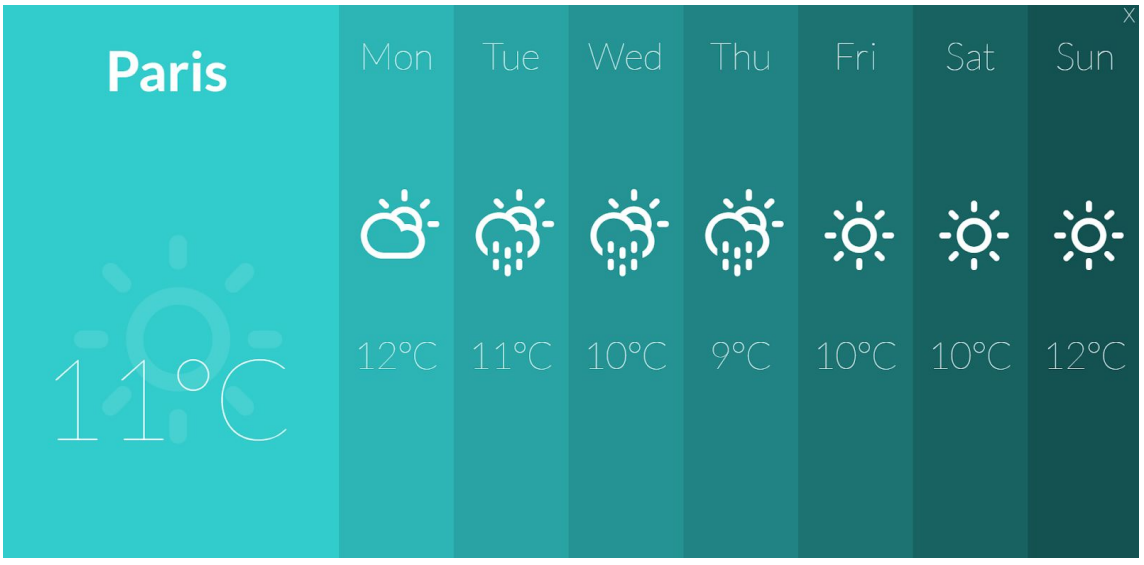
6.5.3 Desenvolupament

Donat que es volia obrir un sol port per la pàgina d'exemples, es va preferir que el document mostrat for un sol document HTML, sense directoris ni reconduccions.

Es va pensar en incloure *slides,* però es va trobar el problema que la gran majoria de les vegades aquestes *slides* requerien javascript. Per aquesta part del projecte, es va voler evitar javascript donat que acostuma a causar diversos problemes i resulta un bon vector d'injecció de codi.

Per tant, es va buscar una plantilla que requerís únicament ús de HTML i CSS. Aquesta plantilla es va trobar també a la pàgina per a creadors digitals Tympanus[16]. Aquesta plantilla consisteix en un únic document HTML amb diversos requadres transparents, els quals ocupen tota la pantalla i es fan opacs al ser pitjats amb el ratolí.



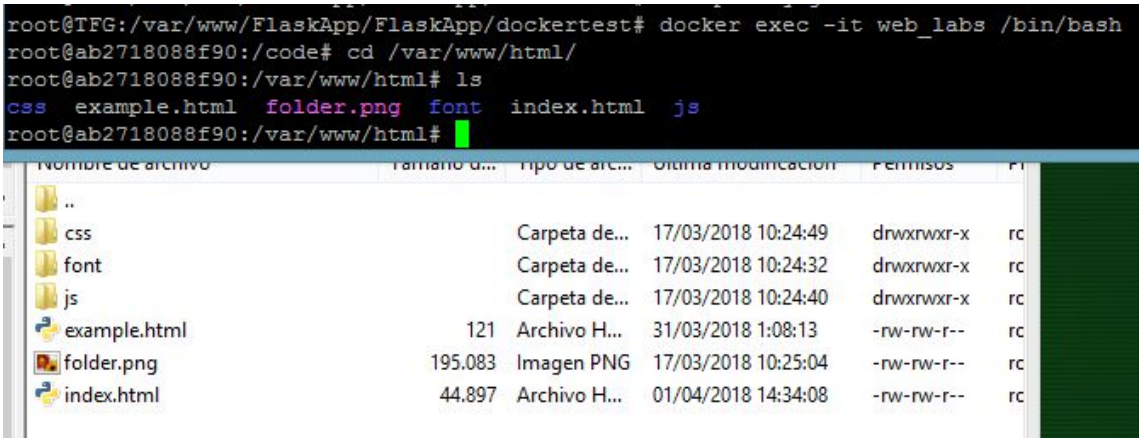


En la imatge anterior es mostra el funcionament de la plantilla que es va escollir. En la primera imatge no s'ha pitjat res, en la segona s'ha pitjat la casella de París i s'ha pres la captura quan s'estava expandint i a l'última

casella es mostra el resultat final de la casella totalment opaca i ocupant la pantalla per complert.

A continuació es va incorporar el material que s'havia recopilat durant la fase de recerca. També es van canviar els colors per una paleta verdosa, donat que es volia emular la sensació de terminal antiga, i es van incloure les terminals falses.

Tal com s'ha explicat en el bloc anterior, ja que no hi havia forma de pujar els documents HTML i CSS directament dins de el contenidor de Docker, es van crear les carpetes compartides. D'aquesta forma, es va crear un directori dins del directori de desenvolupament de la màquina inquilina que comunicava directament amb els continguts que el servidor Apache2 mostrava a internet des del contenidor.



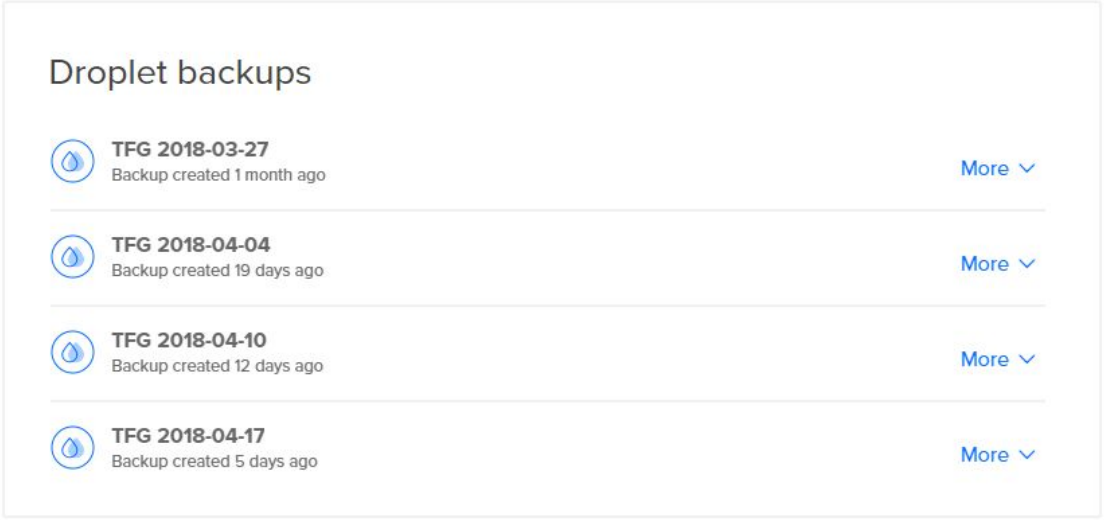
En l'anterior imatge es pot veure com, a dalt, en la terminal, s'ha accedit dins del contenidor i es mostren els continguts que el contenidor mostra a internet. Abaix veiem el programa FileZilla mostrant el directori de la màquina inquilina on es comparteixen les carpetes. Com es pot veure, els continguts de les dues carpetes són els mateixos.

7. Creació de Back-ups

S'ha decidit crear un punt exclusivament per parlar dels back-ups que es van haver de fer del sistema, així com de la seva importància davant de qualsevol projecte. Si no s'hagueren fet, hi hauria hagut danys irreparables en el VPS que haurien causat la necessitat d'una reconstrucció completa i des de zero del projecte.

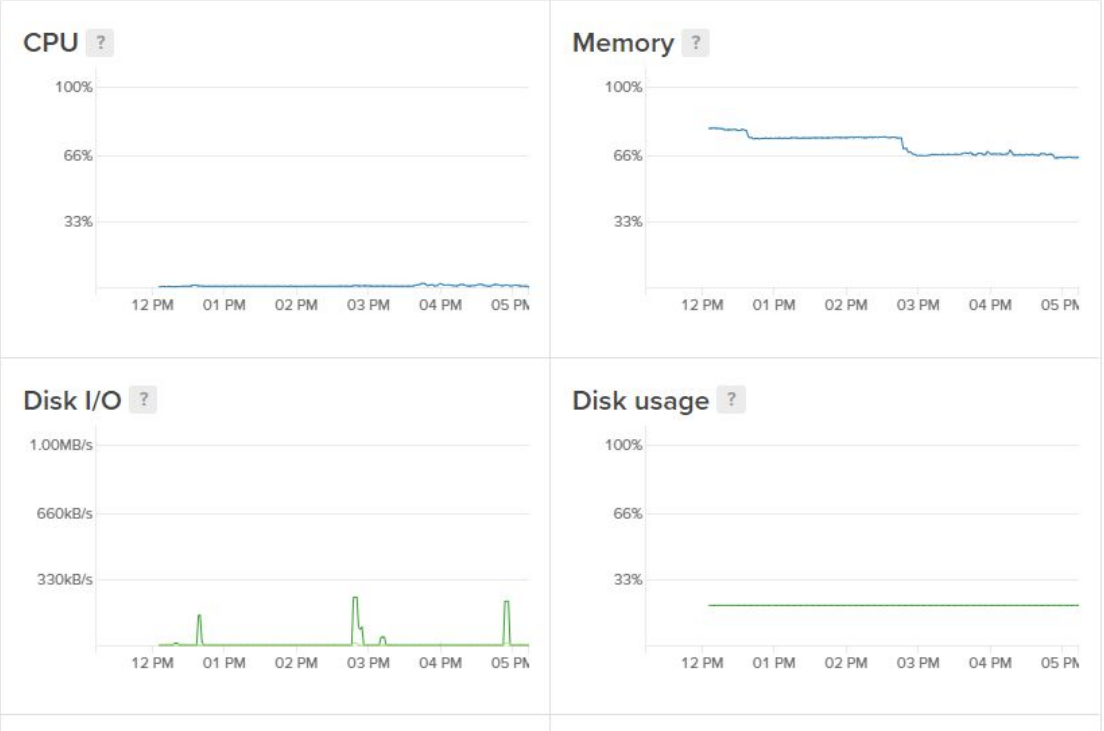
DigitalOcean ofereix, per un dolar més al mes (0.8 cèntims d'euro), la possibilitat de fer back-ups del sistema cada cert temps. Abans de començar res, es va decidir incloure aquesta funcionalitat dins del pack per si passés alguna inconveniència.

Els backups es fan una vegada a la setmana, i es guarden tots els que s'han fet durant el període d'un mes.



Fins al moment, s'ha hagut de fer ús de back-ups en dues ocasions.

En la primera ocasió, es va sobrecarregar el sistema del VPS al crear, destruir, apagar i reiniciar diversos contenidors masses vegades. Això va ser degut a que, en aquell moment, s'estava aprenent el funcionament de Docker i l'ús de les seves comandes, pel qual es va abusar massa d'elles. La màquina sencera es va caure i, amb ella, tots els recursos web i tot el que hi havia en els directoris i fitxers.



CPU i memòria havien sobrepassat els seus límits, i els gràfics es mostraven més elevats del 100%.

Gràcies a que hi havia back-ups disponibles d'uns dies enrere, es va poder tornar a carregar la imatge gairebé igual que com estava abans de caure.

El segon motiu no se sap ben bé perquè va passar, però tot l'HTML, CSS i Javascript que hi havia a la pàgina principal s'havia barrejat amb l'HTML i CSS del contenidor. Se sospita que, ja que es va estar treballant amb el material del contenidor els dies abans, es va pujar per error a la carpeta equivocada i es van sobrescriure alguns documents.

També es va aplicar un back-up del sistema, el qual el va deixar exactament com estava abans.

Es veia necessari parlar dels back-ups del sistema en aquest document ja que moltes vegades no es tenen en compte però, de forma involuntària, han resultat ser uns dels punts més forts i amb més importància d'aquest treball de recerca.

8. Domain name

8.1 Què és i com funciona el DNS?

A continuació es farà una petita introducció al funcionament del DNS per tal que les següents seccions tinguin sentit per al lector.

Per entendre el seu funcionament es procedirà a utilitzar com exemple a seguir la IP y el nom de domini de la pàgina sobre la qual s'ha treballat en aquest projecte.

8.1.1 DNS

En primer lloc, al finalitzar la creació del *droplet* de DigitalOcean, se li va assignar a aquest una IP estàtica. A través d'aquesta IP estàtica un usuari podia accedir a través del navegador.

El mecanisme pel qual el navegador sap que la IP condueix a la pàgina creada és mitjançant la reconducció de la petició que s'ha fet a través dels routers fins a la destinació final, que és el servidor en el qual està allotjat el *droplet*.

Donat que no és pràctic que les persones recordin una sèrie de IPs per anar fins a recursos web, es va decidir introduir el sistema de noms de domini (DNS). Aquest sistema permet convertir el nom en la IP final.

8.1.2 TLDs i dominis

Un nom de domini està conformat de tres o més parts, les quals serveixen per enrutar la petició a través d'internet.

La primera part de totes, és a dir, la que és necessària per poder començar a resoldre i enrutar la petició és el que anomenem el top level domain (TLD). Aquest nivell és resolt pel *Root DNS,* el qual coneix els suficients TLDs per ajudar al navegador a resoldre la petició que està sent formulada.

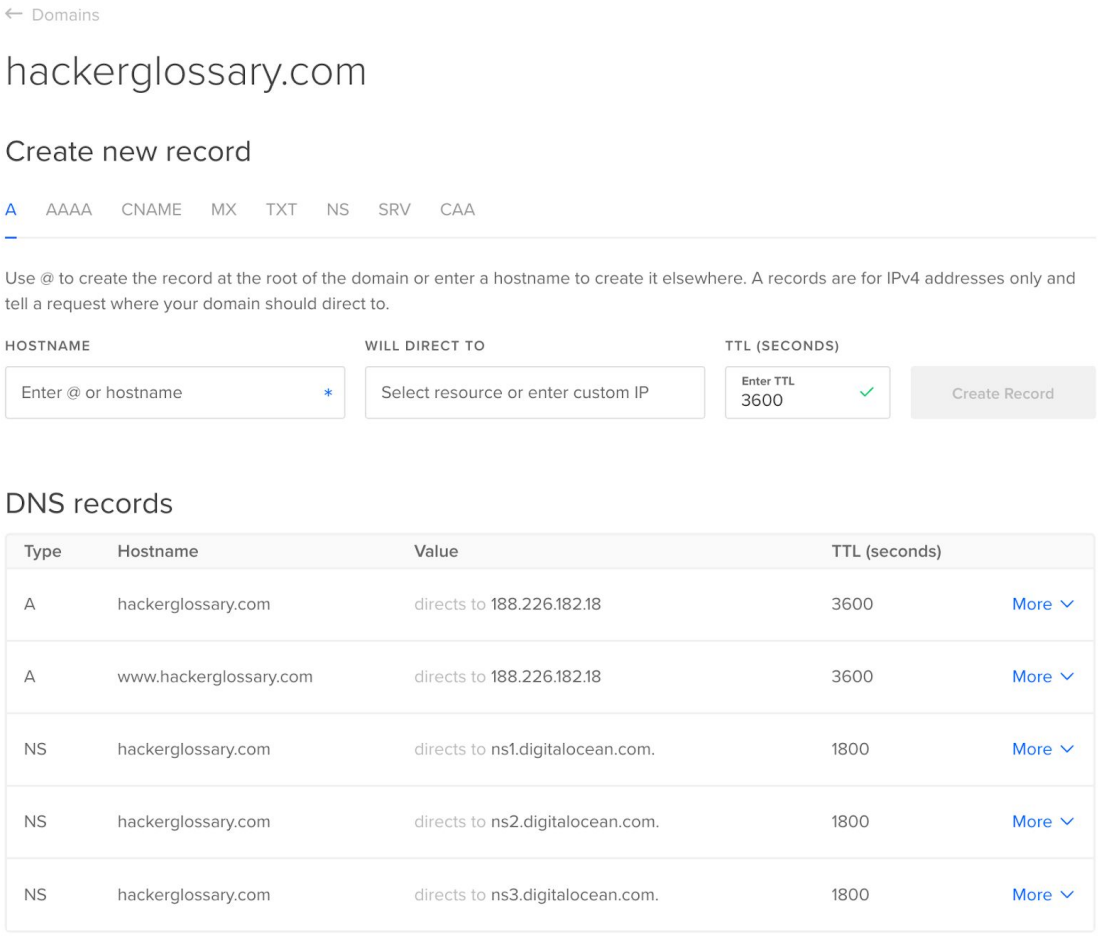
En el nostre cas, el TLD que el *Root DNS* ha resolt ha sigut el conegut .com.

Un cop obtingut el TLD, el següent que serà necessari serà saber a quina organització pertany la ruta a la qual ens dirigim. En el cas presentat, el següent que serà resolt serà el nom hackerglossary.

Un cop aquests dos noms han sigut construïts, es podria seguir fer creixen el DNS en cas de que presentés subdominis. En el present no s'han fet subdominis per la web.

8.1.3 Name servers de DigitalOcean

DigitalOcean presenta la possibilitat de que el dom del domini adquirit apunti als seus servidors propis. El fet de que, amb només *hackerglossary.com* es pugui apuntar cap als servidors propis de DigitalOcean, estalvia molt de temps i de salts entre routers per preguntar on es troben adresses que haurien sigut necessaries sense el nom de domini de DigitalOcean.



8.2 Compra d'un nom de domini en GoDaddy

El domini que es va comprar no venia en DigitalOcean, sinó que s'ha de comprar i enregistrar d'immediat.

Una vegada s'ha pagat pel domini, s'haurà de reclamar d'immediat, ja que el primer en reclamar-lo és qui se'l queda (s'assumeix que qui el declara primer és aquell que l'ha comprat).

El cost va ser de 11 euros.

8.3 Configuració de la xarxa desde DigitalOcean

Un cop es va haver comprat el dom de domini, va procedir-se a apuntar els servidors de DigitalOcean cap a el nom del domini adquirit. També es va apuntar la IP del *droplet* al dom del domini comprat.

Un altre aspecte a aclarir seria quina diferència hi ha entre els DNS records de tipus A i de tipus NS.

8.3.1 Tipus A

Com es pot observar en la fotografia, els servidors de tipus A són els que associen un DNS amb una IP. Per tant, aquest tipus de servidors s'hauran d'utilitzar per associar el nom comprat a través de GoDaddy a una IPv4, la qual és aquella IP que DigitalOcean ha assignat al *droplet* creat.

8.3.2 Tipus NS

El de tipus NS és el que s'encarrega de redirigir una petició DNS cap als servidors de nom autoritaris assignats a DigitalOcean.

Per tant, s'ha assignat el DNS comprat amb els servidors de nom de DigitalOcean, i així quan un usuari realitzi una petició, donat que són noms de servidors coneguts, aquesta petició trigarà menys temps a realitzar-se i no haurà de fer un recorregut complert per la seva resolució.

8.4 Com funciona HTTPS i perquè s'utilitza?

HTTPS és el protocol HTTP servit a través de capes SSL i TLS. SSL es troba ja deprecat, pel que és millor pràctica fer ús del protocol TLS, com s'ha fet servir en aquest cas.

El motiu pel qual es va decidir implementar HTTPS va ser per 2 motius:

- Assegurar-se de que s'està parlant amb el servidor verídic. És a dir, que ningú hagi pogut assumir la identitat del servidor per tal de obtenir les dades que d'altre forma se li proporcionaria a aquest servidor amb el qual s'està interaccionant.

- Assegurar-se de que ningú està fent un atac *Man in the Middle*, o amb altres paraules, que si s'està interceptant la comunicació entre el servidor i l'usuari, aquest atacant no sigui capaç de desxifrar la comunicació encriptada.

Per tal de que servidor i client facin ús del mateix certificat i es puguin desxifrar les peticions i les respostes, és necessari que tots dos estiguin d'acord amb la versió que s'utilitzarà, qui la envia, la clau pública que es farà servir, la signatura i les dates de caducitat.

Un cop corroborades totes aquestes dades, la transmissió segura pot tenir lloc.

8.5 Adquisició d'un certificat

Finalment és va adquirir un certificat per tal d'encriptar les comunicacions HTTP via TLS. El certificat es va obtenir del proveïdor gratuït de *Let's Encrypt.*

9. Adaptació a diferents dispositius

Per tal d'adaptar les pàgines creades a dispositius mòbils, en primer lloc es va valorar si s'haurien d'adaptar o no.

Algunes pàgines no van ser adaptades, ja que per motius estètics no es veien bé en dispositius més petits que un monitor. Donat que és una pàgina web que es fa per compartir informació i no en el món del comerç, se li va donar una gran importància a la part estètica.

Per tant, es van adaptar les pàgines que incloïen informació tècnica, però no les que mostraven exemples.

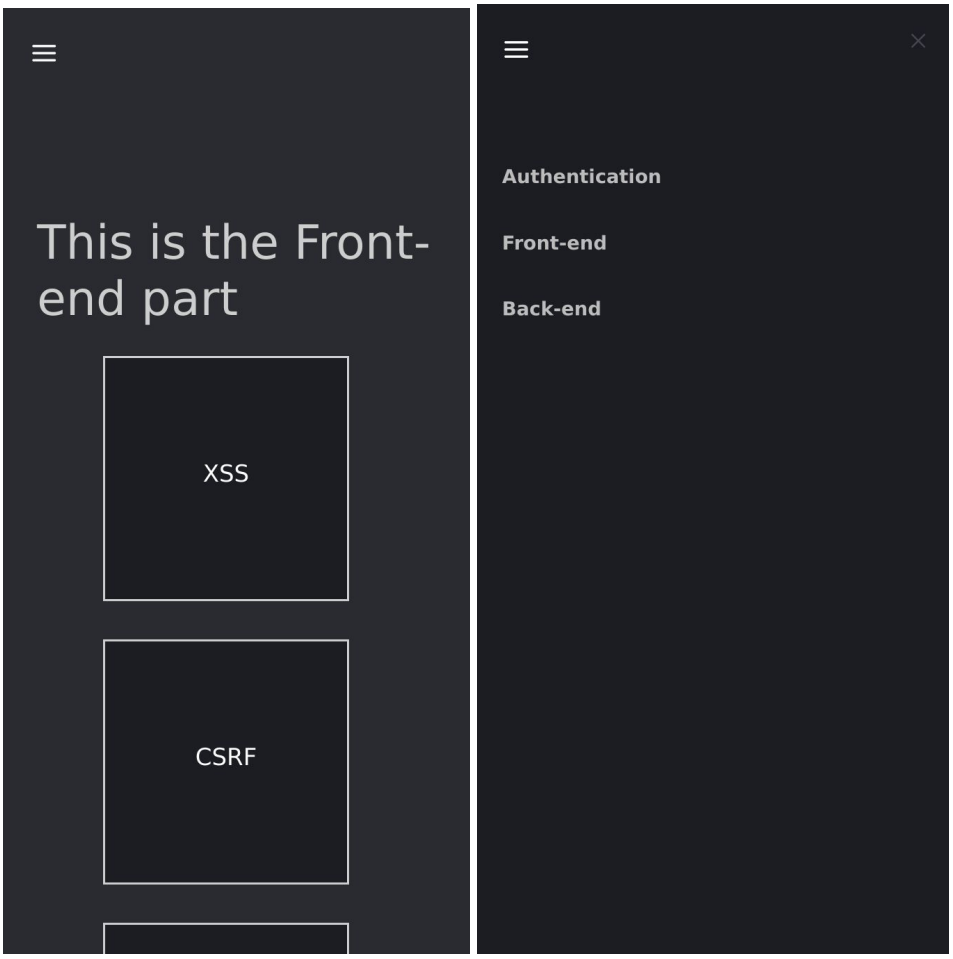
9.1 Adaptació a tamanys més petits

Per tal d'adaptar la pàgina d'informació als tamanys més petits, es va tenir en compte que era important que hi hagués claretat a l'hora de oferir els mecanismes de desplaçament per la pàgina web. Les funcions dels botons havien de quedar clares, així com els apartats.

- Menú desplegable: ja que la pàgina comptava amb un menú de navegació, era vital que l'usuari fos capaç de desplaçar-se i saber on es trobava el menú de navegació en cada moment. Perquè no resultés una navegació confusa, es va optar per oferir un menú

desplegable, ja que és la forma habitual en la qual es presenten els menús en dispositius mòbils.

- Diferents apartats en columna: en comptes de mostrar-se en fila, es mostren els apartats que contenen les dieferents categories en una columna, així l'usuari podrà seguir la metodologia usual de navegació i fer *scroll* per la pàgina.



9.2 Gestió de la no adaptació a tamanys menors

En aquest cas, s'explica perquè s'ha decidit no adaptar la pàgina d'exemples a dispositius mòbils.

- Es vol que la experiència es mantingui en un monitor: la idea inicial era simular que l'usuari estava en una terminal, i per tal la millor forma de simular aquesta experiència és que es mostri en un monitor de tamany real.

10. Bugs en la interfície d'usuari

10.1 Pàgina d'error

Les pàgines d'error són essencials en una pàgina web si es pretén mantenir un nivell de seguretat adequat. El motiu és perquè, sense pàgines d'error, el servidor mateix mostra l'error que ha causat la falta, d'aquesta forma compartint informació sobre el servidor i la versió que s'ha implementat per la pàgina web. Si en algun cas el servidor utilitzat té una falta de seguretat no corregida, l'atacant ha obtingut un punt feble a través del qual explotar la pàgina.

10.1.1 Gestió de les pàgines d'error en el codi

Per tal d'evitar que les pàgines que no existien o provocaven algun error en el sistema poguessin donar informació a atacants, és va crear un mecanisme pel qual, si la pàgina era inexistent, saltava la pàgina d'error.

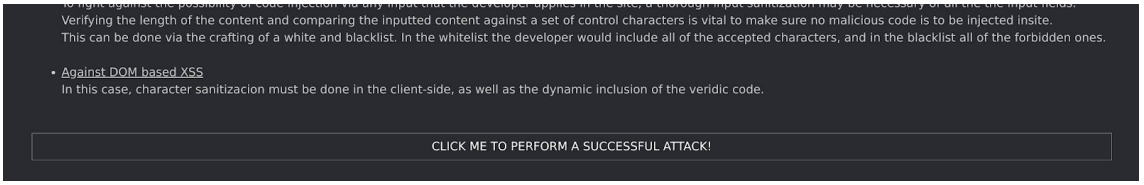
10.1.2 Resultats finals i esperats

Un cop implementat, es va comprovar que, si s'introduia una pàgina inexistent, realment es mostrava la pàgina d'error tal i com es va haver configurat.

10.2 Links no funcionals

Més que un bug, aquest error es tractava d'una falla per part de la desvolupadora, la qual va oblidar-se de fer que el link que portava a les pàgines d'exemple no funcionés.

Va ser fàcil de corregir, ja que només era necessari incloure la URL dins del path. Donat que la pàgina era de contingut dinàmic, només va ser necessari incloure el link una vegada.



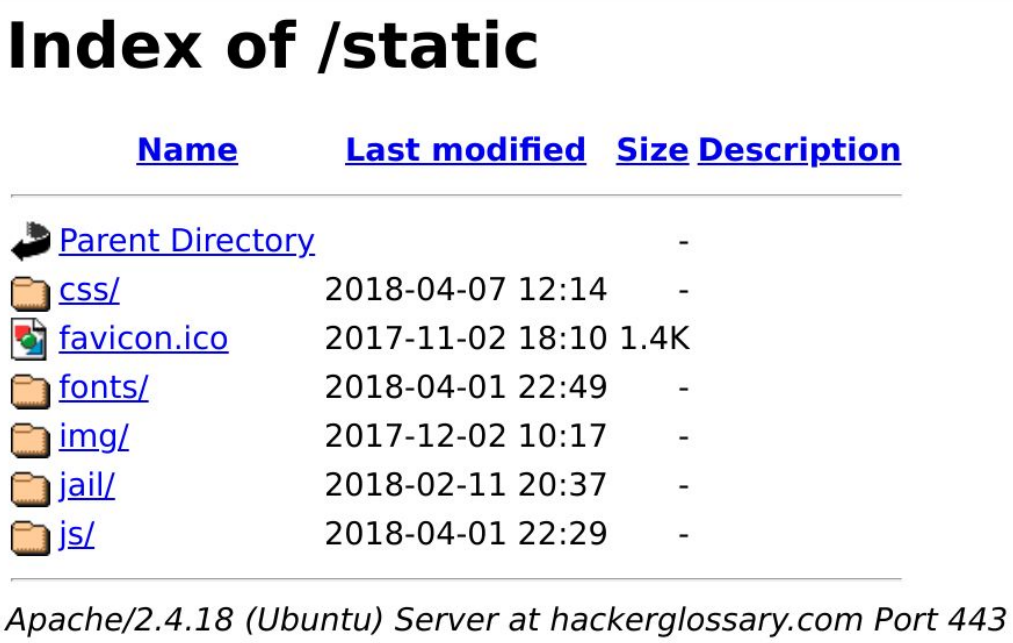
11. Errors de seguretat

El fet de que aquesta pàgina consistís en explicar les falles de seguretat comunes en les pàgines web, no l'eximia de cometre falles de seguretat. Per aquest motiu es va demanar a companys de treball pentesters i amics que també es dediquen de forma professional al món de la ciberseguretat que fessin un anàlisis ràpid a la pàgina web.

El motiu de que no es fes un pentesting real de la pàgina web és que es tracta de tests els quals requereixen diversos dies i són cars.

11.1 Directory listing

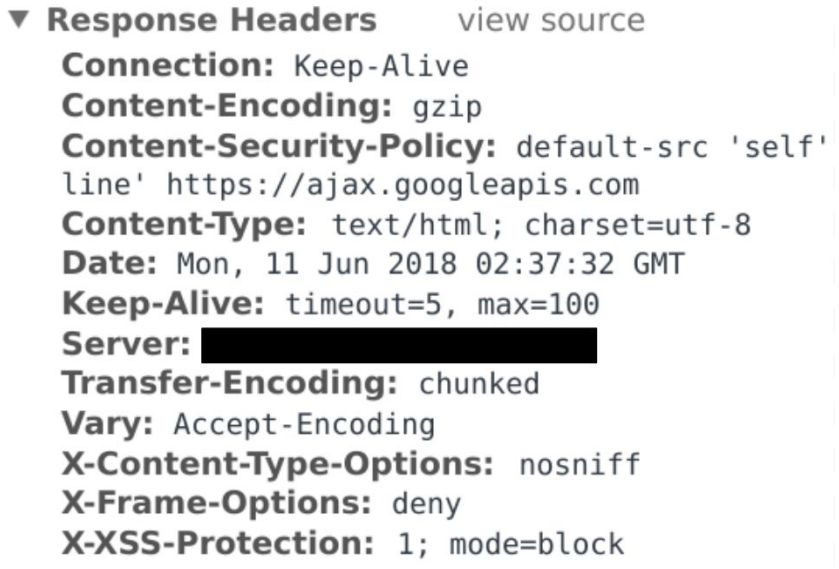
Es va trobar el llistat de pàgines web exposades a l'exterior, lo qual pot provocar que un atacant coneixi la estructura interna del servidor web. Un cop un atacant coneix la estructura web, ja coneix si hi ha fitxers importants oblidats pel desenvolupador, quants directoris de més hi ha per sobre de l'actual i la forma d'anomenar els fitxers que té el desenvolupador.



Per tal de corregir aquest error, és necessari desactivar l'accés a aquestes pàgines.

11.2 Server name disclosure

Un altre forma que un atacant tindria per trobar vulnerabilitats web és mitjançant el nom del servidor que s'està utilitzant.



Si un atacant aconseguís extreure el nom del servidor que s'està fent servir, seria capaç de saber si aquest és vulnerable a algun tipus d'atac conegut. Si és donés la situació de que el servidor emprat és vulnerable, un atacant podria fer-se amb el control de la màquina sencera.

És molt difícil gestionar vulnerabilitats que impliquin l'ús d'un servidor vulnerable, ja que la seva solució implica haver de reimplementar i redissenyar el funcionament intern de la plataforma.

12. AppArmor en Docker

És va buscar la manera d'implementar AppArmor en el contenidor de Docker, i es va trobar que Docker ja carregava un fitxer d'AppArmor per la seva banda.

Es va voler investigar què era exactament el que carregava Docker en els perfils d'AppArmor. La informació que es va extreure del codi original va ser la següent:

- AppArmor especifica els tipus de dispositius de connexió (*mounts*) dintre del sistema que admet, i fora dels tipus indicats no n'admet cap més.

- Bloca la creació de *paths* dintre del sistema que el puguin possar en perill.
- Bloca la possibilitat de que es sobreescriguin certs fitxers o s'alterin directoris delicats.

S'ha cregut que el que s'havia aplicat amb seccomp ja va haver sigut suficient, per tant no s'ha alterat de cap forma.

13. SetUID, SetGUID i Chroot

Aquesta fase del treball va presentar diverses complicacions, les cuals totes naixien de la problemàtica de poder crear un contenidor de Docker sense un usuari root.

13.1 Necessitat de root

Docker és molt similar a una màquina virtual, excepte que comparteix el kernel amb la màquina física. Donat que s'han de crear totes les funcionalitats que conformen una màquina real, amb els directoris, subdirectoris i fitxers, aquesta màquina de Docker ha de tenir permisos absoluts. Als permisos absoluts dins de una màquina de Linux se li anomena permisos de *root*. Per tant, només el *root* podrà crear els directoris, subdirectoris i programes que faran funcionar el sistema operatiu emulat.

13.2 El problema

L'objectiu d'aquest apartat era el de aconseguir fer que una màquina de Docker funcionés sense *root*.

Quan un atacant penetra dins un sistema, un dels objectius més importants és el de fer-se amb el control absolut de la màquina, que en un sistema Linux vol dir tindre tots els permisos o ser *root* del sistema.

Amb tots aquests plantejaments exposats, es va decidir que el millor seria evitar que un atacant pogués ser *root*, per tant, es volia fer que l'usuari per defecte dins del contenidor fos el d'un usuari amb els privilegis el més baixos possibles.

Tot i això, es va veure que no es podia canviar l'usuari per defecte de la màquina. El motiu d'aquest problema era que, al crear-se el contenidor, com que es necessiten tots els privilegis, no és possible canviar-se d'usuari a no ser que s'especifiqui des de fora del mateix contenidor (vector que

es va descartar, ja que si un atacant penetrava dins del contenidor de Docker, el seu vector d'entrada seria el mateix contenidor i no la màquina en la qual s'allotjava).

13.3 Estat actual d'investigació

Fa poc es va descobrir que aquest problema no només existia per aquesta ocasió, sinó que era un problema real que compartien diversos usuaris arreu del món.

Es va descobrir la pàgina https://rootlesscontaine.rs/ i la comunitat https://github.com/opencontainers/runc/issues/1658, en la qual es compartien notícies sobre avenços en la situació actual per aconseguir un contenidor que tingui la possibilitat de ser *rootless.*

Per tant, no es podrà aplicar aquesta solució per aquest projecte.

Tot i això se seguirà investigant i seguint novetats que girin al voltant d'aquest objectiu, contribuint a la comunitat sempre que sigui possible.

14. Utilitat en el món de la ciberseguretat

14.1 Utilitat de Docker en ciberseguretat

Avui en dia l'us de Docker en centres de vigilància digital o SOCs (security operations center) és bastant ampli.

Des d'una perspectiva personal, s'ha presenciat l'us diari, continuat i imprescindible que presentaven les aplicacions que funcionaven sobre les tecnologies Docker.

El motiu d'us és similar al motiu pel qual s'ha volgut utilitzar en aquest projecte: per poder mantenir operacions aïllades de la resta de la xarxa i de les dades principals que s'han guardat a l'ordinador físic. També presenten un mecanisme per evitar que les fallides dels sistemes en diferents contenidors de Docker que gestionen les dades es produeixin alhora; és a dir, per poder aïllar punts d'error. Aquest fet és de suma importància en el negoci de la ciberegureta, ja que el temps que requereix solucionar un problema s'ha de mantenir el més reduït possible.

14.2 Visió de futur

Tot i que el projecte que és presenta ara acaba en aquest punt, es pretén continuar el que es va començar.

La forma en la que es seguirà construint es divideix en dos punts.

- Creant nous apartats dins aquest projecte: en el present s'ha creat aquesta pàgina amb la intenció de mostrar les falles comunes de seguretat en les pàgines web. En un futur, es pretén fer seguir créixer la pàgina principal afegint altres tipus d'errors de seguretat dins altres àmbits, com les xarxes o sistemes criptogràfics.
- Treballant sobre noves funcions de Docker: a mesura que s'avanci en la missió de la comunitat per tal de poder realitzar operacions *rootless* dins d'un contenidor, s'implementaran les noves mesures i millores creades tant per la comunitat com per el repositori principal.

15. Conclusions i treballs futurs

En aquesta última fase del treball s'han repassat els errors tant tècnics com conceptuals que es van fer durant les dues primeres fases del treball.

En quant als errors tècnics, s'ha contemplat l'aspecte de la interacció entre l'usuari i la pàgina, ajustant tamanys i arreglant errors en el disseny. També s'ha contemplat la possibilitat de que un atacant intenti entrar dintre de la pàgina web i prendre el control.

Encara queda treball de futur que aplicar sobre aquesta aplicació web, doncs és un projecte pel qual no s'ha projectat un final, ja que es pretén investigar les novetats i noves mesures que van sortint a partir d'aquesta pàgina web, així com introduir conceptes explorats perquè els nous membres dintre del món de la ciberseguretat puguin ser introduïts a ella.

16. Bibliografia

[1] - Internet Live Stats (2014) http://www.internetlivestats.com/total-number-of-websites/

[2] - Statista (2017) *https://www.statista.com/topics/1145/internet-usage-worldwide/*

[3] - BugCrowd, State of bug bounty report (2017) *https://pages.bugcrowd.com/hubfs/Bugcrowd-2017-State-of-Bug-Bounty-Report.pdf*

[4] - Matt Bishop (2000), University of California: Analysis of the ILOVEYOU worm http://nob.cs.ucdavis.edu/classes/ecs155-2005-04/handouts/iloveyou.pdf

[5] - Robert Graham (2017), RSAConference: Mirai and IoT botnet analysis https://www.rsaconference.com/writable/presentations/file_upload/hta-w10-mirai-and-iot-botnet-analysis.pdf

[6] - OWASP (2001) https://www.owasp.org/index.php/Main_Page

[7] - Flask http://flask.pocoo.org/

[8] - Repositori de Flask https://github.com/pallets/flask

[9] - DigitalOcean https://www.digitalocean.com

[10] - DigitalOcean, How To Install Linux, Apache, MySQL, PHP (LAMP) stack on Ubuntu 16.04 (2016) https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-16-04

[11] DigitalOcean, How To Deploy a Flask Application on an Ubuntu VPS (2013)

https://www.digitalocean.com/community/tutorials/how-to-deploy-a-flask-application-on-an-ubuntu-vps

[12] Docker docs, Get Started https://docs.docker.com/get-started/

[13] DigitalOcean, How To Install and Use Docker on Ubuntu 16.04 (2016) https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-16-04

[14] Docker docs, Create a base image https://docs.docker.com/develop/develop-images/baseimages/

[15] Windows shell, Fabien Gréard https://codepen.io/FabienGreard/pen/mArpBz

[16] Expanding Overlay Effect, Mary Lou (2013) https://tympanus.net/codrops/2013/01/17/putting-css-clip-to-work-expanding-overlay-effect/